July 2016

Extending Faceted Search to the Open-Domain Web

Weize Kong

EXTENDING FACETED SEARCH TO THE OPEN-DOMAIN WEB

A Dissertation Presented

by

WEIZE KONG

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2016

College of Information and Computer Sciences

EXTENDING FACETED SEARCH TO THE OPEN-DOMAIN WEB

A Dissertation Presented

by

WEIZE KONG

Approved as to style and content by:

James Allan, Chair

W. Bruce Croft, Member

David Jensen, Member

Michael Lavine, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

To mom and dad

ACKNOWLEDGMENTS

I would like to express my deepest thanks to my advisor, James Allan, for his immense support and guidance throughout my PhD study. I especially thank James for giving me lots of freedom in research, which makes the years of PhD study a very enjoyable journey.

I thank my committee members, Bruce Croft, David Jensen and Micheal Lavine, for their valuable feedback. I also thank Bruce for his influence on our lab that I have enjoyed, David for influencing me in conducting scientific research, Micheal for his inspiration on part of this thesis. I thank all other research scientists who helped me specifically on this thesis work, including Zhicheng Dou from Microsoft Research Asia, David Smith, Henry Field, Limi Yao and Bo Jiang from UMass Amherst.

I would like to thank all other mentors and friends who helped me in my past research career. I especially thank Yiqun Liu for introducing me to research when I was an undergraduate student. I thank Elif Aktolga for her help in my earlier years of the PhD study. I thank Rui Li, Roger Jie Luo and Yi Chang, who mentored me during my time at Yahoo! Labs.

I thank all staffs of our lab, Kate Morruzzi, Dan Parker, Jean Joyce and Glenn Stowell for their incredible support, which makes it possible for me to enjoy research work and not worry about others. I also thank our Graduate Program Manager, Leeanne Leclerc, for her help throughout this PhD program.

I thank all my friends and colleagues in our lab for sharing various ideas about research, and life as well. In alphabetical order: Qingyao Ai, Elif Aktolga, Michael Bendersky, Keping Bi, Jie Bing, Ethem Can, Marc Cartright, Daniel Cohen, Jeffery

Dalton, Van Dang, Laura Dietz, Shiri Dori-Hacohen, Henry Feild, David Fisher, John Foley, Jiafeng Guo, Stephen Harding, Samuel Huston, Ashish Jain, Myung-ha Jang, Jiepu Jiang, Mostafa Keikha, Chang Bong Kim, Jin Young Kim, Youngho Kim, Kriste Krstovski, Chia-Jung Lee, Tamsin Maxwell, Venkatesh Narasimha Murthy, Nada Naji, David Novak, Jae Hyun Park, Jangwon Seo, Manmeet Singh, Ibrahim Uysal, Ziqi Wang, David Wemhoener, Xiaoye Wu, Xiaobing Xue, Zeki Yalniz, Liu Yang, Xing Yi, Xucheng Yin, Hamed Zamani, Michael Zarozinski. I wish them all the best for their future endeavors.

Finally, I would like to thank my parents for their education that encouraged my independent thinking, their support in whatever path I chose to pursue, and their love from the other side of the globe.

This work was supported in part by the Center for Intelligent Information Retrieval, in part by IBM subcontract #4913003298 under DARPA prime contract #HR001-12-C-0015 , in part by an award from Google, in part by NSF grant #IIS-0910884, and in part by UpToDate. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

ABSTRACT

EXTENDING FACETED SEARCH TO THE OPEN-DOMAIN WEB

MAY 2016

WEIZE KONG

B.Eng., BEIHANG UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor James Allan

Faceted search enables users to navigate a multi-dimensional information space by combining keyword search with drill-down options in each facets. For example, when searching "computer monitor" in an e-commerce site, users can select brands and monitor types from the the provided facets {"Samsung", "Dell", "Acer", ...} and {"LET-Lit", "LCD", "OLED", ...}. It has been used successfully for many vertical applications, including e-commerce and digital libraries. However, this idea is not well explored for general web search in an open-domain setting, even though it holds great potential for assisting multi-faceted queries and exploratory search.

The goal of this work is to explore this potential by extending faceted search into the open-domain web setting, which we call Faceted Web Search. We address three fundamental issues in Faceted Web Search, namely: how to automatically generate facets (facet generation); how to re-organize search results with users' selections on

facets (facet feedback); and how to evaluate generated facets and entire Faceted Web Search systems.

In conventional faceted search, facets are generated in advance for an entire corpus either manually or semi-automatically, and then recommended for particular queries in most of the previous work. However, this approach is difficult to extend to the entire web due to the web's large and heterogeneous nature. We instead propose a query-dependent approach, which extracts facets for queries from their web search results. We further improve our facet generation model under a more practical scenario, where users care more about precision of presented facets than recall.

The dominant facet feedback method in conventional faceted search is Boolean filtering, which filters search results by users' selections on facets. However, our investigation shows Boolean filtering is too strict when extended to the open-domain setting. Thus, we propose soft ranking models for Faceted Web Search, which expand original queries with users' selections on facets to re-rank search results. Our experiments show that the soft ranking models are more effective than Boolean filtering models for Faceted Web Search.

To evaluate Faceted Web Search, we propose both intrinsic evaluation, which evaluates facet generation on its own, and extrinsic evaluation, which evaluates an entire Faceted Web Search system by its utility in assisting search clarification. We also design a method for building reusable test collections for such evaluations. Our experiments show that using the Faceted Web Search interface can significantly improve the original ranking if allowed sufficient time for user feedback on facets.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

There are primarily two search paradigms in use since the very beginning of web search, navigational search and direct search. Navigational search uses a hierarchy structure (taxonomy) to enable users to browse the information space by iteratively narrowing the scope of their quest in a predetermined order, as exemplified by Yahoo! Directory[1] and the Open Directory Project[2]. Taxonomies provide a guided search interface, and support abstractions that are easily understood by users. However, the strict ordering of a taxonomy can be too rigid, especially for large and heterogeneous corpora. The rapid decline of Yahoo! Directory as a primary web search engine provides pragmatic evidence (Sacco and Tzitzikas, 2009).

Direct search instead allows users to specify their own queries as input, resorts to search systems for understanding search intents behind user queries, and returns search results that could best address the search intents. This approach has been made enormously popular by web search engines, such as Google[3]. However, in the basic search interface, users have to formulate their queries with no or limited assistance, and no exploration capability since results are presented as a flat list with no systematic organization. Recent advances, including query suggestions and search result clustering, address part of these problems and will be reviewed in Chapter 2.

---

[1]http://en.wikipedia.org/wiki/Yahoo!_Directory

[2]http://www.dmoz.org/

[3]http://www.google.com

A third approach that combines the two search paradigms emerged during the 1990s, namely faceted search. More specifically, faceted search enables users to navigate a multi-dimensional information space by combining direct search with narrowing choices in each dimensions, which are also called facets. For example, consider looking for a computer monitor in an e-commerce site like Amazon[4]. Users can directly search with the query "*computer monitor*". However, the number of computer monitors retrieved could be overwhelming. To assist users in refining and exploring the search results, the site provides refining choices in each facets. In this case, the facets are computer monitor attributes, such as "*brand*" with the choices {"*Dell*", "*ViewSonic*", "*HP*", ...}, "*display technology*" with the choices {"*LED-Lit*", "*LCD*", ...}, and "*condition*" with the choices {"*new*", "*used*", "*refurbished*"} , as illustrated in Figure 1.1. Then users can select some of the provided choices and combine them to refine the search results. For example, users can select the choice "*new*" from the facet "*condition*" to request only computer monitors in new conditions. Users can also combine the choice "*Dell*" in the facet "*brand*" with the choice "*LCD*" in the facet "*display technology*" to request only Dell computer monitors with LCD display technology.

Compared with direct search, faceted search provides additional search assistance for users. The facets provided in faceted search can assist users in clarifying their search intent and refining the search results, as already illustrated in the computer monitor example above. In addition, faceted search supports guided exploration over the complex information space. In the computer monitor example above, without the provided facets, users might have no clue about how to explore the large amount of returned results. By listing computer monitor attributes as facets, the site gives users an overview of the returned results, and provides them with the key factors they may need to consider when searching for a computer monitor. In other words, using facets,

[4]http://www.amazon.com

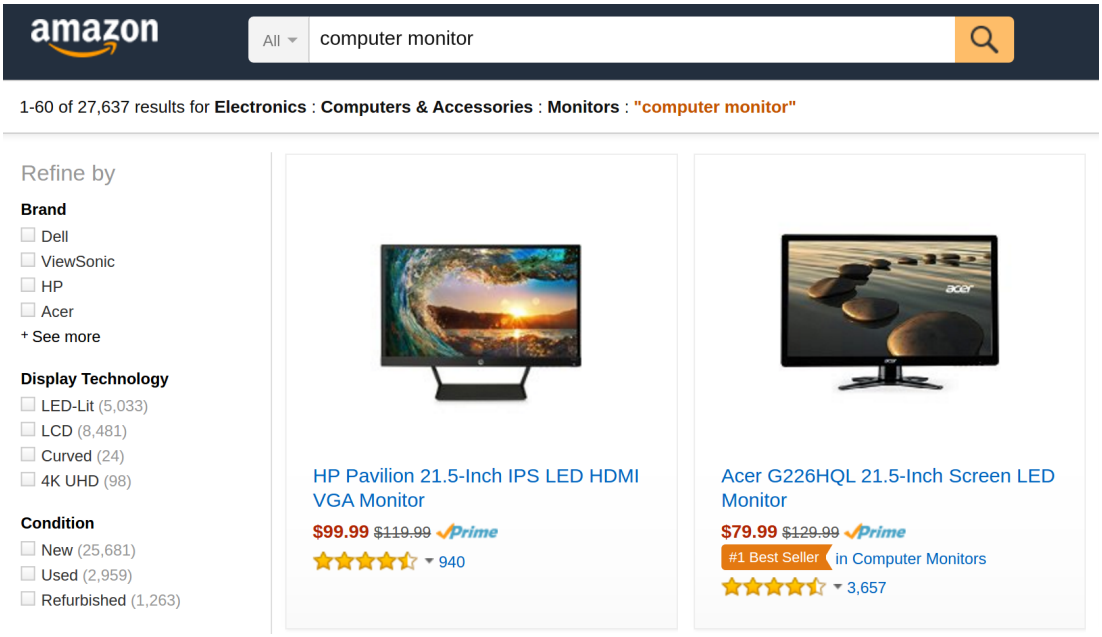


Figure 1.1: Faceted search illustrated by an example searching "computer monitor" in Amazon. The search interface shown has been simplified for the convenience of illustration.

the site summarizes the search space succinctly, and provides exploration suggestions organized in an systematic way. This exploration capability is especially important in exploratory search tasks, or when users are not exactly clear about what they are looking for.

Compared with navigational search, faceted search is different not only in its direct search capability, but also in its navigation mechanism. Navigation in navigational search is typically based on one single taxonomy that organizes information objects in a hierarchy structure. The key property of such a taxonomy is that, for every object in the taxonomy, there is precisely one unique path to it from the taxonomy root. For example, "*cancer*" can be classified in a taxonomy with a unique path "*disease*"→"*structural disease*"→"*tumor*"→"*cancer*". However, this strict ordering of a taxonomy can be too rigid when dealing with compound information objects. For example, should "*treatment of cancer*" be a child of "*treatment*" or of "*cancer*"? This strict ordering constraint limits expressibility and extensibility of taxonomy.

Instead, in faceted search, navigation is based on multiple independent taxonomies, called faceted taxonomies or multi-dimensional taxonomies. Each of the taxonomies organizes information objects from a different (preferably orthogonal) point of view, or equivalently in a different dimension of the multi-dimensional information space. The independent taxonomies in a faceted taxonomy can also be called "*facets*". However, "*facets*" are often used to indicate the part of independent taxonomies that are shown to users, and in many cases the taxonomies are often shown as shallow trees. In the computer monitor example above, the facets show only two-level taxonomies (*e.g.*, node "*Brand*" with children "*Dell*", "*ViewSonic*", etc.). We will provide a more detailed explanation of facets and related concepts in Chapter 2.

Combining these independent taxonomies, a faceted taxonomy offers expressive power and flexibility beyond a single taxonomy used in navigational search. Ranganathan first proposed the idea of faceted taxonomy in library science. In *Classification, Coding, and Machinery for Search* (Ranganathan, 1950), he provided an example that expresses the topic "*statistical study of the treatment of cancer of the soft palate by radium*" based on four constituent taxonomies in a faceted taxonomy as follows.

- Medicine → Digestive system → Mouth → Palate → Soft palate
- Disease → Structural disease → Tumor → Cancer
- Treatment → Treatment by chemical substances → Treatment by a chemical element → Treatment by a group 2 chemical element → Treatment by radium
- Mathematical study → Algebraical study → Statistical study

In the example, the four taxonomies ("*Medicine*", "*Disease*", "*Treatment*" and "*Mathematical study*") each represents one dimension of the information space. The foci in each of the taxonomies ("*Soft palate*", "*Cancer*", "*Treatment by radium*" and "*Statisti-*

cal study") are combined to express the compound topic, which is difficult in a single hierarchical taxonomy.

In summary, faceted search combines direct search with navigation based on a faceted taxonomy, providing assistance for users in clarifying search intent and exploration over a multi-dimensional information space. It has also been used successfully for many vertical applications, such as e-commerce and digital libraries.

1.1 Motivation for Faceted Web Search

While the principles of faceted taxonomy are widely applicable, faceted search has not been explored much for general web search in an open-domain setting. There is lots of work studying faceted search in fixed domain such as images (Cutrell et al., 2006), movies (Koren et al., 2008), houses (Shneiderman, 1994), and desktop content (Cutrell et al., 2006). However, there is limited work in successfully extending faceted search to the open-domain, due to the challenges of the large and heterogeneous nature of the web (Teevan et al., 2008).

Nevertheless, faceted search naturally suits and holds great potential for assisting multi-faceted queries and exploratory search in the open-domain web setting. We illustrate the idea with four example queries and their facets manually created in Table 1.1. For the first query "*baggage allowance*", the facet of different airlines (facet 1) can assist users to quickly compare baggage policies between different airlines. Users can also use other facets, such as flight types, flight classes and specifications to clarify this multi-faceted query. For the second query, "*Mars landing*", the facets list important information for exploring this topic, including Mars rovers (facet 1) and countries relevant to Mars landing (facet 2). Users can also combine multiple facets for search intent clarification (e.g., select "*Curiosity*" from facet 1 and "*pictures*" from facet 3 to find pictures of Mars rover Curiosity). The last query shows that faceted search is not limited to short queries. The facets succinctly summarize interesting

aspects for the complex query "*Effect of building the three gorges dam in China*", including different types of effect (facet 1), related nature disasters (facet 2) and different regions of the river (facet 3). This information could help users to learn and explore this topic.

Table 1.1: Examples facets for three web search queries.

Query 1: baggage allowance
Facet 1: AA, Delta, Jetblue, ...
Facet 2: international, domestic
Facet 3: first class, business class, economy class
Facet 4: weight, size, quantity
Query 2: Mars landing
Facet 1: Curiosity, Opportunity, Spirit
Facet 2: USA, UK, Soviet Union
Facet 3: video, pictures, news
Query 3: Effect of building the three gorges dam in China
Facet 1: environmental, social, economic
Facet 2: landslides, soil erosion, earthquake
Facet 3: lower, middle, upper

Faceted search when extended to the open-domain web setting can have several advantages over other related techniques developed for faceted queries or exploratory search. Search result diversification has been studied as a method of tackling ambiguous or multi-faceted queries while a ranked list of documents remains the primary output feature of web search engine today (Agrawal et al., 2009; Clarke et al., 2008; Santos et al., 2010; Sakai and Song, 2011; Dang and Croft, 2013). The purpose is to diversify the ranked list to account for different search intents or query subtopics. However, the query subtopics are hidden from the user, leaving him or her to guess at how the results are organized. Faceted search addresses this problem by explicitly presenting different facets for the search topic.

Search results clustering or organization is a technique that tries to organize search results by grouping them into clusters (Cutting et al., 1992; Käki, 2005; Zamir and Etzioni, 1999; Carpineto et al., 2009) or organizing them in a single hierarchical

taxonomy (Lawrie et al., 2001; Lawrie and Croft, 2003; Nevill-Manning et al., 1999), very much like in navigational search. This offers a complementary view to the flat ranked list of search results. However, single taxonomies or fixed clusters, as mentioned before, impose strict taxonomic ordering for the search results. Instead, faceted search allows users to explore the search space from multiple facets, offering flexibility beyond search results clustering or organization.

Query suggestion (Baeza-Yates et al., 2004; Cao et al., 2008) is a technique that generates alternative queries for users to helps them explore or express their information need. This technique is now widely used in commerce web search engine. However, the suggested queries are often provided in a flat list with no systematic organization and no capability of combining multiple suggestions. Instead, "suggestions" in faceted search are organized systematically into facets, which makes it conceptually easier to browse and select.

1.2 Faceted Web Search

In this thesis, we extend faceted search into the open-domain web setting, which we call Faceted Web Search (FWS). Similar to conventional faceted search, a FWS system will provide facets when a user issues a web search query. The user can then select some choices from the facets, which will be used by the FWS system to adjust the search results to better address the user's information need.

We illustrate the idea of FWS in Figure 1.2, supposing a user is preparing for an international flight and wants to find baggage allowance information. When the user searches "baggage allowance" in an FWS system (step 1 in the figure), in addition to the search result list (step 2), the system will provide a list of facets (step 3), such as a facet for different airlines, {"*Delta*", "*JetBlue*", "*AA*", ...}, a facet for different flight types, {"*domestic*", "*international*"}, and a facet for different classes, {"*first*", "*business*", "*economy*"}. When the user selects choices such as "*Delta*", "*international*"

and "*economy*" in these facets (step 4), the system can ideally help to bring web documents that provide baggage allowance information for the economy class of Delta international flights to the top of the search results (step 5).

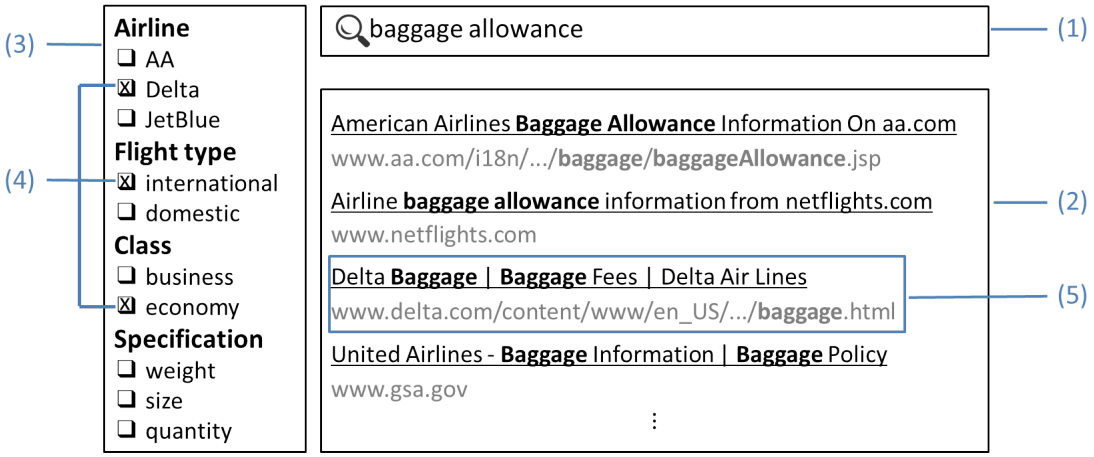


Figure 1.2: An example of Faceted Web Search: (1) the user issues a query; (2) the system returns search results; (3) the system provides facets for the query; (4) the user selects terms in the facets; (5) the system re-ranks this relevant document to the top according to the selected terms. Note that our work does not generate labels for facets (*e.g.*, "*airline*", "*flight type*").

This thesis address three fundamental issues of Faceted Web Search, including facet generation, facet feedback, and evaluation, as described further as follows.

1.2.1 Facet Generation

Facet generation is to identify facets for navigation (corresponding to step 3 in Figure 1.2). In conventional faceted search, facets are generated in advance for an entire corpus (Stoica and Hearst, 2007; Dakka and Ipeirotis, 2008) either manually or semi-automatically, and then recommended for particular queries in most of the previous work (Teevan et al., 2008). However, this approach is difficult to extend to the entire web due to the web's large and heterogeneous nature. We instead propose a query-dependent approach, which extracts facets for a query from its search results, providing a promising direction for solving the problem.

We call the extracted items "query facets". A query facet is a set of terms (*e.g.*, {"*AA*", "*Delta*", "*JetBlue*",...}) that are subsumed by an implicit label (*e.g.*, "*airlines*"). The set of terms succinctly represents different options in the same category (*e.g.*, "*airlines*") that a user can select to refine the issued query (*e.g.*, "*baggage allowance*"). More examples for query facets are shown in Table 1.1. Note that our work dose not generate the implicit label (*e.g.*, "*airlines*") for the set of terms. This definition of query facets corresponds to one-level taxonomies in faceted taxonomy, in which only information objects that belong to a same parent node are shown as a facet (see the definition in Section 2.4). We leave generating facets as two- or more level taxonomies as future work.

Because it is an automatic task, facet generation in FWS can be imperfect. The system can make mistakes in both precision and recall for generating facets. As in many precision-oriented information retrieval tasks, we believe users are likely to care more about "facet precision" than "facet recall". That is, users may care more about the correctness of presented facets (*e.g.*, are the terms in the airline facet indeed about airlines, and are the airline terms grouped together in a same facet) than the completeness of facets (*e.g.*, are all possible facets for that query presented, and are all possible airline terms included the results?). In other words, mistakes of presenting wrong terms in a facet, or grouping terms incorrectly are more severe than omitting some facets or terms in facets. Thus, we also study how to improve facet generation performance under precision-oriented scenarios, in order to make the technique more practical.

1.2.2 Facet Feedback

Facet feedback is to use selections on the facets to adjust (*e.g.*, filter or re-rank) the search results (corresponds to step 5 in Figure 1.2). In conventional faceted search, facet feedback is straightforward: as all information objects have been classified in

the faceted taxonomy, when users make their selections on the facets (called feedback terms), the search results can be easily filtered by requiring each objects belong to the restricted taxonomies according to the selection.

However, in FWS there is no explicit classification of webpages into the generated facets. One solution is Boolean filtering, which filters search results by the requiring selected terms to appear. However, it turns out to be too strict when extended to the open-domain setting. Boolean filtering is based on two assumptions (Zhang and Zhang, 2010): (1) users are clear about what they are looking for, and thus are able to select proper terms to restrict the results; and (2) matching between a term and a document is accurate and complete. In FWS, that means a document that contains the selected term should be relevant to the term, and all documents relevant to that selected term should contain the term. Neither of the two assumptions are likely to hold reliably in FWS. Thus, we also investigate soft ranking models that expand original queries with user selection on the facets.

1.2.3 Evaluation

Evaluation for conventional faceted search mostly focuses on its user interface (Burke et al., 1996; English et al., 2002; Hearst, 2006a, 2008; Kules et al., 2009). For FWS, we study two types of evaluation according to the different focuses, namely intrinsic and extrinsic evaluation. Intrinsic evaluation only considers facet generation (i.e., the quality of generated facets). Extrinsic evaluation instead evaluates the effectiveness of the entire FWS system, combining both the facet generation and facet feedback components.

Most of the previous evaluations for faceted search are based on user studies (Dash et al., 2008; Li et al., 2010; Stoica and Hearst, 2007). However, user studies are often very expensive and more importantly difficult to extend for evaluating new systems.

We instead design evaluation methods with higher reusability for both intrinsic and extrinsic evaluation.

In the following, we highlight the contributions of this thesis.

1.3 Contributions

- Faceted Web Search. We define Faceted Web Search, which extends faceted search into an open-domain web setting. We design a framework for an FWS system, which contains the facet generation and facet feedback components. We show that using this faceted search interface can significantly improve the original ranking if allowed sufficient time for user feedback: 18.0% in NDCG@10 if we allow users to examine 50 terms in facets, and 7.4% in NDCG@10 if we allow time for examining 10 terms. We also find that the skip list structure (Section 7.3) in Faceted Web Search interface can help users save time in considering feedback terms in irrelevant facets, and achieve higher re-ranking performance when users are looking for more feedback terms, comparing to a term relevance feedback model based on RM3 (Abdul-Jaleel et al., 2004; Lavrenko and Croft, 2001).

- Query facet extraction. To cope with the large and heterogeneous nature of the web in facet generation, we develop a query-dependent approach, which generates facets for a query instead of the entire corpus. This query facet generation approach extracts facets from the top search results for the issued query. This not only makes the generation problem easier, but also addresses the facet recommendation problem at the same time. For query facet extraction, we develop a supervised approach based on a graphical model to recognize facets from the noisy candidates found. The graphical model learns how likely a candidate term is to be a facet term as well as how likely two terms are to be grouped together in a query facet, and captures the dependencies between

the two factors. We propose two algorithms (QFI and QFJ) for approximate inference on the graphical model since exact inference is intractable. Compared with other existing methods, our models can easily incorporate a rich set of features, and learn for available labeled data.

- Intrinsic evaluation. We evaluate the quality of generated facets by comparing them with human-created ones. This can be measured from three aspects – precision and recall of extracted terms for facets, and the clustering quality of these facet terms. We design $PRF_{\alpha,\beta}$, a measure to combine three evaluation factors together using weighted harmonic mean. This metric has the flexibility to adjust emphasis between the three factors for different applications. We also describe how to collect human annotations for query facets by a pooling method. Experimental results based on this intrinsic evaluation show that our supervised methods (QFI and QFJ), can take advantage of a richer set of features and outperform other unsupervised methods, such as pLSA, LDA, and a variant of quality threshold clustering model (Dou et al., 2011). Our experiments also identify several informative features (Section 4.4.5) and important extraction patterns (Section 4.4.6) for query facet extraction.

- Precision-oriented query facet extraction. We improve query facet extraction performance under precision-oriented scenarios from two perspectives. First, we find that the likelihood objective used in the query facet extraction model can be loosely related to the performance measure in the precision-oriented scenario. Therefore, we directly optimize the performance measure instead of likelihood during training using a empirical utility maximization approach. However, exact optimization on the performance measure is difficult due to the non-continuous and non-differentiable nature of information retrieval measures. We address this problem by approximating the performance measure using its expectation. We

show that this empirical utility maximization approach significantly improves models under precision-oriented scenarios, suggesting that utility is a better learning objective than likelihood, and that our expectation-based approximation is effective.

- Second, we improve extraction performance by a selective method that shows facets for good performing queries and avoids doing so for poor performing ones. We find that extraction performance varies for different queries – some queries are naturally more difficult than others for extracting query facets. In the precision-oriented scenario, it may be more desirable to avoid showing facets for those poor performing queries and leave the users with a clean keyword-search interface. A key problem, however, is how to predict the extraction performance. To solve this problem, we develop a simple and effective score based on the expectation of the performance measure. We find the score has a strong correlation with the performance measure, and when used in the selective method, it can significantly improve the average performance with fair coverage over the whole query set.

- Facet feedback. We find that Boolean filtering models are too strict for FWS, and propose soft ranking models that expand original queries with user selected terms in facets for re-ranking. We show that the proposed soft ranking models are more effective than Boolean filtering models, which are widely used in conventional faceted search.

- Extrinsic evaluation. We develop an extrinsic evaluation method that evaluates FWS systems by their utility in assisting search clarification. This evaluation considers both gain in ranking improvement and cost in time for users to give feedback. Instead of performing user studies, we simulate the user feedback process, so that we can easily extend the evaluation for new models or systems.

The simulation is based on a simple user model of the feedback process and limited human annotations.

- Building a reusable test collection. We describe a way of building reusable test collections for the intrinsic and extrinsic evaluation. We make our collected data set publicly available. The data set consists of annotated facets for 196 TREC Web Track queries from 2009 to 2012, and simulated user feedback for 678 corresponding query subtopics.

1.4 Outline

The remainder of this thesis is organized as follows. In Chapter 2, we provide background information related to this thesis. In Chapter 3, we present our query facet extraction approach. In Chapter 4, we present our intrinsic evaluation that evaluates generated facets by comparing them with human-created ones. In Chapter 5, we investigate query facet extraction models under precision-oriented scenarios, and improve our models in such scenarios. In Chapter 6, we investigate both Boolean filtering and soft ranking models for facet feedback. In Chapter 7, we develop our extrinsic evaluation method that evaluates entire Faceted Web Search systems in terms of their utility in assisting search in an interactive search task. Lastly, in Chapter 8, we summarize the contributions made in this thesis and discuss potential future directions for more research in this area.

CHAPTER 2

BACKGROUND

In this chapter, we discuss background information and related work for Faceted Web Search, an extension of faceted search in the open-domain web setting. Faceted search is a heavily interdisciplinary area, where different aspects of information retrieval, knowledge representation and human computer interaction must be considered all together. Therefore, we first describe related topics in these areas as a background for faceted search/Faceted Web Search. Then, we describe faceted search, Faceted Web Search and previous research on these topics. After that, we discuss related approaches that aim to achieve the same goals as our work. We defer discussion of some related work to later chapters where the context makes it more appropriate.

2.1 Knowledge Representation

Faceted search is built upon taxonomies and faceted taxonomies, which are two types of knowledge representations that haven been studied for many years.

2.1.1 Taxonomy

The word "*taxonomy*" originally referred to the classification of biological organisms. Its history dates back to more than two millennia ago. At that time, the Greek philosopher Aristotle first classified all living things into a hierarchical classification system, a taxonomy (Tunkelang, 2009). The taxonomy classified living things by dividing them into two groups, plants and animals; further dividing animals into those "red blood" and "no red blood"; those with no red blood into "hard bodies" and "soft bodies"; and so forth (Figure 2.1).

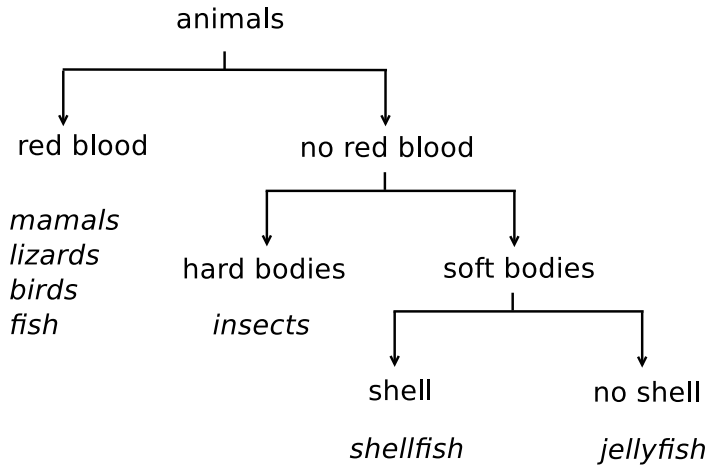


Figure 2.1: A subset of Aristotle's knowledge taxonomy (Tunkelang, 2009)

Today, the word "*taxonomy*" refers more generally to any hierarchical classification schema. Tunkelang (2009) described a taxonomy as an organization of things or abstractions into a hierarchy or tree structure. Similarly, Sacco and Tzitzikas (2009) described a taxonomy as a concept hierarchy going from the most general to the most specific concepts, into which information objects can be classified. Using Aristotle's taxonomy as an example, we can see that concepts or abstractions of animals are organized in a tree. The root node "*animals*" corresponds to the set of all animals. Its children represent the top-level divisions of the animals, one representing "*red blood*" animals and one representing "*no red blood*" animals. Their children correspond to the subdivisions of those animals; and so forth. Information objects are classified directly into concepts in the tree. For example, the object "*jellyfish*" is classified to the concept "*no shell*" and the object "*insects*" is classified to the concept "*hard bodies*".

To provide a more clear definition for a taxonomy, we synthesize its previous definitions and formulations (Sacco and Tzitzikas, 2009; Tzitzikas et al., 2005), and define a taxonomy as follows:

Definition 2.1.1. A **taxonomy** is a tree of concepts with all concepts subsuming their descendant concepts.

Next we explain the terminology used in this definition. First, informally, a **tree** (like a real but upside-down tree) has a single root node at the top, leaf nodes at the bottom, and branches connecting each nonleaf parent node to its children. (See Garnier and Taylor (2009) for a formal definition of a tree.) Figure 2.1 shows an example of tree. The root node is "*animals*", and the parent nodes have edges pointing to their children. **Descendants** of a node A are nodes under A's branch. For example, Descendants of "*no red blood*" include "*hard bodies*", "*soft bodies*", "*shell*" and "*no shell*". A tree of concepts are simply a tree using concepts as nodes.

Second, a **concept** in a taxonomy is an abstraction which identifies all the information objects classified under it (including the information objects that are classified to its descendants in the concept tree). For example, in Figure 2.1, the concept "*no red blood*" identifies all "no-red-blood" animals in the taxonomy, including "*insects*", "*shellfish*", "*jellyfish*" that are classified to its descendants. More formally, a **concept** C can be defined as a set of information objects $C = \{d\}$. However, before being materialized with information objects, a concept is just an abstraction for a set of potential information objects. Also, concepts are often presented by their textual labels to convey the concept meaning to users.

Last, in the definition, the concept tree is constrained by having concepts in the tree subsume all their descendant concepts. A concept A is **subsumed** by a concept B ($A \preccurlyeq B$) if the set of information objects classified under A is intentionally constrained to be equal to or a subset of the set of objects classified under B: $A \subseteq B$. For example, in Figure 2.1, we can say that the concept "*shell*" is subsumed by the concept "*no red blood*", because every animals classified under "*shell*" is also classified under "*no red blood*". This definition of subsumption indicates a reflexive and transitive binary relation over concepts. It is very general, and thus can model many different reflexive

and transitive binary semantic relations, such as IS-A and PART-OF. For example, the subsumption relation "*shell*" $\preccurlyeq$ "*no red bold*" indicates a "shell" animal is a "no-red-blood" animal. For PART-OF relations, an example is a taxonomy of United States locations in which the concept "*USA*" has a child node "*MA*", and the concept "*MA*" has a child node "*Amherst*". Thus, we have "*MA*" $\preccurlyeq$ "*USA*", "*Amherst*" $\preccurlyeq$ "*MA*" and "*Amherst*" $\preccurlyeq$ "*USA*" (from the transitive property). Here the subsumption relations between these locations model PART-OF relations (*e.g.*, "*MA*" is a part of "*USA*").

Taxonomies provide a systematic way for organizing all types of knowledge or information. This idea, which originated from Aristotle's work, influences different science fields today. In library science, Melvil Dewey developed a taxonomy for organizing books in the 1870s, called Dewey Decimal Classification (DDC) (Dewey, 1876), which is still used by many library today. In computer science, perhaps the most familiar example of taxonomies is the web directory that Yahoo! built in the mid-1990s, which organizes web sites into hierarchical categories (Van Couvering, 2008).

The hierarchical tree organization of information objects in a taxonomy enables efficient navigation through the information space, and provides the basis for navigational search (described in Section 2.2.1). Users start from the root node and iteratively discriminate among children, in order to search for the appropriate one. Each time a node is selected for expansion, the total number of objects to be considered is reduced because the objects classified under discarded concepts need not be considered. Thus, users iteratively reduce the number of information objects to be manually inspected (Sacco and Tzitzikas, 2009).

The key property of the tree structure in a taxonomy is that, for every information object or set of objects that corresponds to a node, there is precisely one unique path to it from the root node. Thus, a taxonomy imposes a strict logical ordering on the information that it represents (Tunkelang, 2009). For example,

"*cancer*" can be classified in a taxonomy with a unique path "*disease*"→"*structural disease*"→"*tumor*"→"*cancer*". However, this strict ordering of a taxonomy can be too rigid when dealing with compound information objects. For example, should "*treatment of cancer*" be a child of "*treatment*" or of "*cancer*"? This strict ordering constraint limits expressibility and extensibility of taxonomies and navigational search systems build on them.

A workaround for the limitation is to use a polyhierarchy instead of a tree structure for taxonomy. In a polyhierarchy taxonomy, a node may have multiple parents – and thus multiple different paths leading to it from the root. Polyhierarchy introduces additional expressibility. For example, in a polyhierarchy, "*treatment of cancer*" can be assigned as a child node of "*treatment*" and a child node of "*cancer*" at the same time. However, the introduction of polyhierarchy creates more problems than it solves, particularly for maintaining polyhierarchy taxonomies. It is difficult enough to maintain a standard taxonomy, and the difficulty becomes much greater when moving a node does not simply move its subtree with it (Tunkelang, 2009).

In the next section, we describe faceted taxonomy which provides a more elegant solution to the limitation of a single hierarchical taxonomy.

2.1.2 Faceted Taxonomy

The idea of faceted taxonomy (sometimes also called multi-dimensional taxonomy (Sacco and Tzitzikas, 2009) or faceted classification (Tunkelang, 2009)) is to combine multiple independent taxonomies for describing or organizing compound information objects. This idea was first introduced in library science by Shiyali Ramamrita Ranganathan. Ranganathan saw the limitation inherent in using a single hierarchical taxonomy to represent a diverse collection of books. To solve the problem, he developed the colon classification scheme in 1933 (Ranganathan, 1933). The colon classification scheme is based on multiple independent taxonomies. It expresses a

compound object as a sequence of symbols (letters and numbers) separated by colons (and thus it was named colon classification). Each separation of the symbol sequence represents the foci in one of the independent taxonomies. To given an example, Ranganathan (1950) describes such a compound object "*the statistical study of the treatment of cancer of the soft palate by radium*", represented by *L2153:4725:63129:B28*. This compound object can be broken down into four independent taxonomies:

- Medicine (L) → Digestive system (L2) → Mouth (L21) → Palate (L215) → Soft palate (L2153)
- Disease (4) → Structural disease (47) → Tumor (472) → Cancer (4725)
- Treatment (6) → Treatment by chemical substances (63) → Treatment by a chemical element (631) → Treatment by a group 2 chemical element (6312) → Treatment by radium (63129)
- Mathematical study (B) → Algebraical study (B2) → Statistical study (B28)

The foci ("*Soft palate*", "*Cancer*", "*Treatment by radium*" and "*Statistical study*") in each of the taxonomies are combined to express the compound object, which is difficult in a single hierarchical taxonomy.

To give a definition for a faceted taxonomy, we follow Tzitzikas et al. (2005) and define a faceted taxonomy as:

Definition 2.1.2. A **faceted taxonomy** is a set of independent taxonomies.

Next we explain the term "*independent*" in the definition. Informally, each independent taxonomy organizes information objects from a different point of view, or equivalently based on a different dimension of the multi-dimensional information space. Formally, a taxonomy A is **independent** of a taxonomy B if A and B do not have any common concepts. In order words, a concept cannot appear in multiple independent taxonomies. This ensures the decomposition of a faceted taxonomy (into

disjoint taxonomies). Each taxonomies can thus be extended independently, and if the concept or tree structure of one taxonomy is changed, other taxonomies would not be affected (Wei et al., 2013).

Navigation based on a faceted taxonomy is called **faceted navigation**. In faceted navigation, users can use multiple taxonomies to navigate or explore a multi-dimensional information space. They can narrow the search space by iteratively selecting child nodes in each of taxonomies. Then the selections on the taxonomies are combined to restrict the matched information objects. This has been illustrated in the computer monitor example in Figure 1.1. In the example, users can combine taxonomies "*brand*", "*display technology*" and "*condition*" to restrict the computer monitor search results.

We summarize the major advantages of a faceted taxonomy as follows. Overall, by combining multiple independent taxonomies, faceted taxonomies offer expressive power and flexibility beyond a single taxonomy. More specifically, first, a faceted taxonomy can easily express compound information objects by combining multiple independent taxonomies. For example, Ranganathan expressed the complex topic "*the statistical study of the treatment of cancer of the soft palate by radium*" by combining four independent taxonomies "*medicine*", "*disease*", "*treatment*", "*mathematical study*". Second, each independent taxonomy in a faceted taxonomy can be extended and updated easily without affecting other taxonomies. Using Ranganathan's example, adding new type of mathematical study in the "*mathematical study*" will not affect "*medicine*", "*disease*" or "*treatment*".

2.1.3 Other Knowledge Representation

Besides taxonomy and faceted taxonomy, there are other types of knowledge representation including thesaurus and ontology. We briefly discuss thesaurus and ontologies here, since they are not the focus of this work. A **thesaurus** is a set of

terms plus a set of relations between these terms (Jing and Croft, 1994). These relations may include BT (Broader Term), NT (Narrow Term), UF (Used For) and RT (Related To). Examples for thesauri include WordNet, LCSH (Library of Congress Sub ject Headings) and MeSH (Medical Subject Headings). Similar to a taxonomy, a thesaurus may organize terms in a hierarchal tree. However, a thesaurus could also capture associative and equivalent relations between the terms in addition to the hierarchal relation. An **ontology** is a set of entities and the relationships among those entities (Gruber, 1993). Compared with taxonomies and thesauri, ontologies allow for more general relationship types. For example, we may have "*Paris*" is-capital-of "*France*" or "*Barack Obama*" is-president-of "*USA*". A comparative analysis of taxonomy, thesaurus and ontology is provided by Kumar (2013).

2.2 Search Paradigms

Faceted search combines two search paradigms, direct search and navigational search.

2.2.1 Navigational Search

Navigational search (sometimes also called "*directory navigation*") uses a taxonomy to enable users to browse the information space by iteratively narrowing the scope of their quest in a predetermined order (discussed in Section 2.1.1). Examples include Yahoo! Directory[1] and the Open Directory Project[2]. Navigational search provides a guided search interface, and supports abstractions that are easily understood by users. However, the strict ordering imposed by the hierarchy structure in the taxonomy can be too rigid, especially for large and heterogeneous corpora (Snow et al., 2006; Tunkelang, 2009; Sacco and Tzitzikas, 2009). The rapid decline of Yahoo! Direc-

[1] http://en.wikipedia.org/wiki/Yahoo!_Directory

[2] http://www.dmoz.org/

tory as a primary web search engine provides pragmatic evidence. One solution to the problem is to use **faceted navigation** . As discussed in Section 2.1.2, faceted navigation is based on a faceted taxonomy instead of a single taxonomy. With a faceted taxonomy, users can combine multiple taxonomies to navigate a multi-dimensional information space.

2.2.2 Direct Search

Direct search instead allows users to specify their own queries as input, and is the dominant paradigm in the field of information retrieval. The queries are often keywords in web search scenarios, and thus, sometimes direct search is also called "*keyword search*". Direct search resorts to search systems for understanding search intents behind user queries, and returns search results that could best address the search intents. This approach has been made enormously popular by web search engines, such as Google[3]. However, in the basic search interface, users have to formulate their queries with no or limited assistance, and no exploration capability since results are presented as a flat list with no systematic organization. Faceted search aims to solve this problem, as discussed in Section 2.3. We also discuss other recent advances for addressing this problem in Section 2.5.

2.3 Faceted Search

Faceted Search combines direct search with faceted navigation, which enables users to navigate a multi-dimensional information space. In Figure 2.2, we repeat the computer monitor example used in Chapter 1. In the example, a user searches with the query "*computer monitor*" in Amazon (direct search), and the site provides multiple *facets* ("*brand*", "*display technology*" and "*condition*") for users to select and

[3]http://www.google.com

narrow down the search results (faceted navigation). Next we explain what are facets.

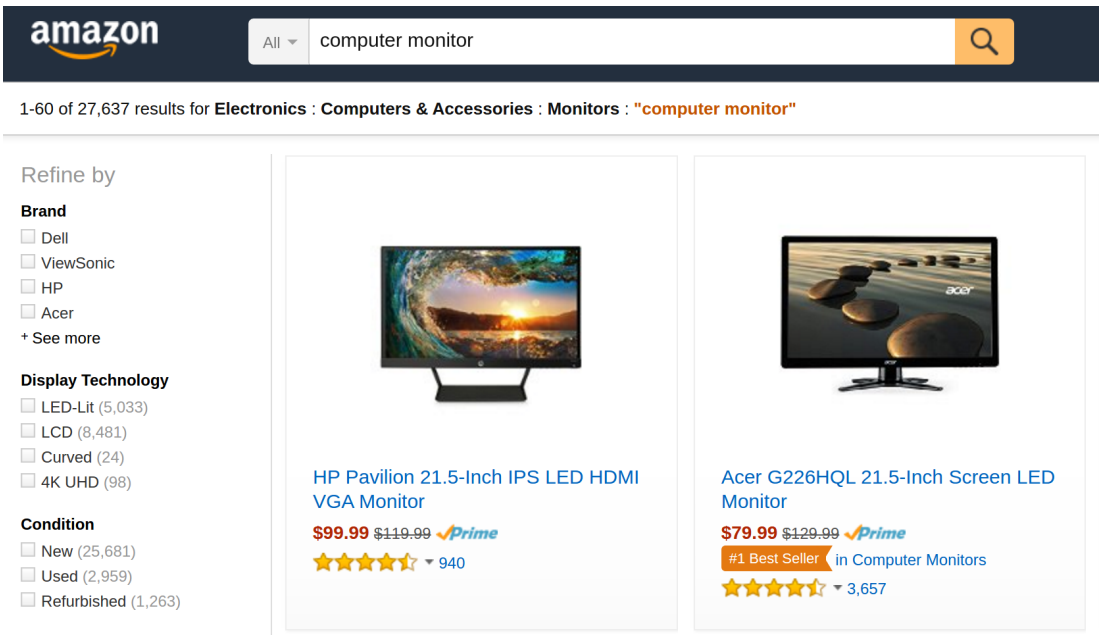


Figure 2.2: Faceted search illustrated by an example searching "computer monitor" in Amazon. The search interface shown has been simplified for the convenience of illustration.

2.3.1 Facets

In general, the term "*facet*" means "little face" and is often used to describe one side of a many-sided object, especially a cut gemstone. In information science literature, "*facet*" is a term that are often introduced with less-formal definitions. For example, Hearst (2006a) described facets as categories used to characterize information items in a collection. Koren et al. (2008) described facets as metadata that can define alternative hierarchical categories for the information space. Bonino et al. (2009) defined a facet as an independent point of view for representing the content of a resource. Facets are sometimes also called "attributes", "dimensions", and "faceted metadata" in other literature (Teevan et al., 2008; Li et al., 2010; Yee et al., 2003).

We provide a more precise definition for facets under the context of a faceted taxonomy as follows:

Definition 2.3.1. Facets are the independent taxonomies in a faceted taxonomy.

Note that this definition is built on our definitions for a taxonomy (Definition 2.1.1) and a faceted taxonomy (Definition 2.1.1). We synthesize them for facets as follows. First, a facet is a taxonomy, which is a tree of concepts with all concepts subsuming their descendants. Second, facets are independent from each other in the faceted taxonomy – there are no common concepts in multiple facets, or informally, each facet organizes information objects from a different point of view. In Ranganathan's example (Section 2.1.2), the four independent taxonomies "*medicine*", "*disease*", "*treatment*", "*mathematical study*" in the faceted taxonomy are facets. In practice, facets are often built as (or shown as) shallow or two-level trees. In the computer monitor example (Figure 2.2), the facets are two-level (*e.g.*, parent node "*Brand*" with child nodes "*Dell*", "*ViewSonic*", "*HP*", "*Acer*").

We describe facets under the context of Faceted Web Search in Section 2.4.

2.3.2 Comparison with Other Search Paradigms

Compared with direct search, faceted search provides additional search assistance for users through facets. Facets can assist users in clarifying their search intent and refining the search results (*e.g.*, select "*new*" in facet "*condition*" to find only computer monitors in new condition). Facets also summarize the search space succinctly, and provide exploration suggestions organized in a systematic way (*e.g.*, the listed facets "*brand*", "*display technology*", and "*condition*" give users an overview of the returned results, and provide them with the key factors they may need to consider when searching for a computer monitor). This exploration capability is especially important in exploratory search tasks, or when users are not exactly clear about what they are looking for (Kules et al., 2009; Sacco and Tzitzikas, 2009).

Compared with navigational search, faceted search is based on a faceted taxonomy that enables faceted navigation, which, as outlined above, is especially useful for navigating a multi-faceted information space. In navigational search, the strict ordering imposed by a taxonomy is too rigid when dealing with compound information objects in the multi-faceted information space. Instead, in faceted search, users can combine facets to express a complex information need (*e.g.*, "*statistical study of the treatment of cancer of the soft palate by radium*" in Ranganathan's example).

Next we discuss previous work on faceted search from four aspects – user interface, facet generation, facet recommendation and evaluation.

2.3.3 Faceted Search User Interface

Thought not the focus of our work, the user interface is an important factor for faceted search and attracts lots of research attention. The goal is to design a faceted search interface that help users to make effective uses of facets – a user interface that supports flexible faceted navigation with directed search and at all times retaining a feeling of control and understanding.

From a more general perspective, for users to make effective use of search refinement options, the refinements must offer users what Pirolli et al. (2000) call **information scent**: cues that indicate to users the value, cost of accessing the refinement options. In the context of faceted search, it is thus important that users are provided with some indicators about the effect for accessing each facets or facet terms. This is typically accomplished by displaying facet terms with the counts of matched search results for each terms. For example, in Figure 1.1, the facet "*display technology*" and "*condition*" are shown together with the numbers of matched computer monitors for each facet terms (*e.g.*, 25,581 for "*new*" condition, and 2,959 for "*used*" condition). Ben-Yitzhak et al. (2008) investigated a more sophisticated method that returned not only counts of the matched search results for facets, but also richer aggregations that

could support more effective decision making. For example, when shopping for books and looking to refine the search by facet "*author*", it might make more sense to drill down to the author who wrote the most bestsellers, instead of focusing on the author who wrote the most books. So they showed not only the counts of books matched for each authors, but also the numbers of best sellers for them.

The Flamenco project led by Marti Hearst was the most visible research project focusing on faceted search, especially for the user interface aspect of it. The goals of the Flamenco project were to investigate how to assist navigation and browsing of information collections via the use of facets or other hierarchical taxonomies. They addressed questions such as how to allow the user to navigate in several hierarchies simultaneously, how to show facets and matched search results, how to display the query as it is built up, how to present the query previews, and so on.

Hearst (2006a) provided a nice summary of the best practices in user interface design for faceted search based on 13 years of experience in experimenting with Flamenco and other systems. For example, they investigated whether users would be comfortable navigating in multiple facets simultaneously (English et al., 2002). Their usability experiments showed that a strong majority of participants preferred being allowed to navigate in multiple facets, and they felt they were in control and did not feel lost. They studied how to expose hierarchal facets without crowding the display or confusing users. For Flamenco, they adopted a step-by-step drill-down approach in which the level in the facet just below the current selected level is visible, along with a trail indicating the higher level labels. In addition, when the mouse hovers over a label, its immediate children are displayed in a tooltip, so the user can in facet see three levels simultaneously. This approach discloses the facet hierarchy progressively, allowing users to see results grouped by higher-level concepts in order to obtain an understanding of the contents of the collection.

2.3.4 Facet Generation

In most of the previous work, facets are generated manually or can be derived directly from the structured data (Basu Roy et al., 2008; Yee et al., 2003; Dash et al., 2008; Ben-Yitzhak et al., 2008). For example, in e-commerce sites such as Amazon[4], products are stored as structured data with manually created schema. Each product is attached with related attributes such as "*brand*", "*price*", "*condition*", etc. These attributes can be directly used as facets for these products. Similarly, Basu Roy et al. (2008) used 19 attributes (*e.g.*, "*actor*", "*genre*") as facets for a movie databases, and used 43 attributes (*e.g.*, "*made*", "*model*", "*mileage*") as facets for a car database. Dash et al. (2008) used publication attributes (*e.g.*, "*author*", "*venue*", "*time*", "*location*") as facets for the DBLP corpus[5]. As discussed before, since the web is large and heterogeneous, it is infeasible to generated facets or build such databases for it manually.

Previous work that automatically generates facets (or a single taxonomy) is typical based on some existing thesauri or knowledge bases, such as WordNet (Stoica and Hearst, 2007; Dakka et al., 2005; Dakka and Ipeirotis, 2008; Latha et al., 2010) and Wikipedia (Dakka and Ipeirotis, 2008; Li et al., 2010; Kohlschütter et al., 2006). The idea is to leverage the hierarchical structures already built in these resources.

Stoica and Hearst (2007) created taxonomies for a corpus consists of 13,000 recipes using the hierarchical structure base on the hypernym (IS-A) relations in the WordNet (Fellbaum, 1998). They first selected a subset of terms that are intended to best reflect the topics in the documents. Then they created taxonomies using the selected terms by mapping into the WordNet, and leveraging the IS-A relations in the WordNet to build hierarchical structure of these selected terms.

---

[4]http://wwww.amazon.com

[5]http://dblp.uni-trier.de/

Dakka and Ipeirotis (2008) generated facets for news corpora based on both WordNet and Wikipedia. They first selected important terms (or keywords) in the news articles as facet terms. The important terms were found by identifying named entities or Wikipedia titles appearing in the news articles, as well as resorting to Yahoo Term Extraction web service[6], which takes a text document as input and returns a list of keywords for the document. In their pilot study, they found labels for the facet terms usually do not appear in the news articles. Thus, they used hypernyms in WordNet as well as link structures in Wikipedia to assign facet labels for the selected facet terms.

Li et al. (2010) build a faceted search system for Wikipedia by inducing taxonomies from the existing category hierarchy and link structures in it. Since the Wikipedia webpages have already been classified into the existing category hierarchy by human editors, they can directly use these categories as facets, and rank them at query time.

The previous work described above shows that existing hierarchically-structured resources, like thesaurus and knowledge bases, can be useful for automatic facet generation. However, using thesaurus and knowledge bases also restricts the extensibility of these facet generation methods to the general and open-domain settings. Thus most of the work applied the methods only in a domain-specific setting or only for copora that have direct associations in the knowledge bases. For example, the faceted search system (Li et al., 2010) built based on Wikipedia are only applied for the Wikipedia corpora itself, because the facet generation approach requires the documents being directly associated with categories in Wikipedia.

Our work instead is for a more general setting. It target automatic facet generation for the open-domain web. This is recognized as a challenging and unsolved problem (Tunkelang, 2009; Wei et al., 2013; Teevan et al., 2008). Our solution is

[6]http://developer.yahoo.com/

different from previous work in its query-dependent nature. In most of the previous work, facets are generated in advance for an entire corpus (Stoica and Hearst, 2007; Dakka and Ipeirotis, 2008) either manually or semi-automatically, and then recommended for particular queries. We instead propose a query-dependent approach, which extracts facets for a query from its search results. This not only makes the generation problem easier (because the search result data is smaller and less heterogeneous than the entire web corpus), but also addresses the facet recommendation problem at the same time.

2.3.5 Facet Recommendation

Facet recommendation is to select or rank facets (already generated) that are useful for a given query, to be presented in the facet interface for the user. There are two motivations for facet recommendation. First, there may be a large number of facets in a faceted search system, and not all of them are relevant to the search task the user is current in. Second, even if the facets are relevant to the current search task, there may be too many of them or the user interface may be too small to display them all.

In some domain-specific settings, the faceted search system only need support a relatively narrow range of search tasks or search intents (Teevan et al., 2008). In this case, it is easy to predict which facets will be the most useful for the user. For example, in the case of e-commerce sites, facets like "*price*" and "*brand*" may be particularly useful. In the case of recipe search, facets like "*ingredients*" or "*course*" may be most useful.

Previous work has also studied ranking facets automatically. The basic idea is to estimate the usefulness of a facet based on how relevant the facet is to the query (or representative for the given data set) and how efficient it is for navigation. Wei et al. (2013) provide an extensive review on this topic. In the following, we briefly

describe one work as an example. Oren et al. (2006) ranked facets based on predicate balance, object cardinality and predicate frequency. Predicate balance is the balance of the taxonomy tree of a facet. This is based on the idea that tree navigation is most efficient when the tree is well balanced. Thus, the predicate balance is an indicator for navigation efficiency. Object cardinality is the number of child nodes assigned to the facet (its root). When there are too many choices (too many child nodes), the options are difficult to display and the choice might confuse the user. Thus, smaller object cardinality may indicate more efficient navigation. The predicate frequency is the frequency of the facet been assigned to the documents in the corpus. They assumes that a good facet should be applied frequently in the corpus, because if a facet occurs infrequently, selecting a child node from it would only affect a small subset of the documents.

In open-domain settings, the queries applied to a faceted search are varied in intent, and thus it is infeasible to pre-defined sets of facets for recommendation. There is also very limited work in facet recommendation for automatically generated facets under the open-domain settings, because the problem of automatic facet generation is still unsolved. As mentioned before, our query-dependent approach tries to solve both facet generation and facet recommendation in the open-domain setting at the same time. Our approach extracts facets from the search results. We assumes the search results are relevant to the user's search intent, and therefore the facets extracted from them should also be related to the user's search intent.

2.3.6 Faceted Search Evaluation

Compared to the mature evaluation methodology for information retrieval systems, evaluation of faceted search is still nascent (Wilson and m.c. schraefel, 2007; Tunkelang, 2009). Most evaluations for faceted search are based on user studies and focus on the user interface aspect (Burke et al., 1996; English et al., 2002; Hearst,

2006a; Yee et al., 2003; Hearst, 2008; Kules et al., 2009). For example, English et al. (2002) investigated whether users would be comfortable navigating in multiple facets simultaneously (English et al., 2002). Their user study showed that a strong majority of participants preferred being allowed to navigate in multiple facets, and they felt they were in control and did not feel lost. Similarly, via usability studies, Yee et al. (2003) show that when incorporated into a properly-designed user interface, hierarchical facets provides a flexible, intuitive way to explore a large collection of items that enhances feelings of discovery without inducing a feeling of being lost.

Most of the evaluations for facet generation and facet recommendation are also based on user studies (Li et al., 2010; Stoica and Hearst, 2007). For example, Li et al. (2010) conducted user studies to evaluate the effectiveness of generated facets for Wikipedia. They showed facets generated by different methods to the human subjects, then the subjects were administered a questionnaire asking questions, such as "*which interface is better than the other?*", "*what is your rating about usefulness of the interface*". The questionnaire results were aggregated for evaluating generated facets. Stoica and Hearst (2007) conducted user studies for their automatic facet generation approach, and found that their system achieves higher quality results than other automated category creation algorithms, and 85% of the study participants said they would like to use the system for their work.

While user studies provide a direct evaluation for faceted search, they are typically expensive, and more importantly difficult to extend for evaluating new systems or methods. We instead design evaluation methods for faceted search that are much easier to be reused for evaluating new systems. More specifically, we design intrinsic evaluation for evaluating the generated facets themselves by comparing them with ground-truth facets (Chapter 4). We also design extrinsic evaluation for evaluating the entire Faceted Web Search systems based on simulated search tasks (Chapter 7).

Both evaluations are not based on user studies, and can be used relatively easily for evaluating new models or systems.

2.3.7 Faceted Search Systems

Next we briefly discuss a few research and commercial faceted search projects. More extensive discussions for them are provided by Tunkelang (2009) and Wei et al. (2013).

Perhaps the first well-known faceted search project is Flamenco (FLexible information Access using MEtadata in Novel COmbinations) developed by Marti Hearst and her colleagues (Hearst, 2000). Its centerpiece is an open-source faceted search system that supports hierarchical facets. The project represents almost a decade of work on developing faceted search tools and performing usability studies with them. More specifically, Hearst and her colleagues have conducted research on faceted search user interface (Hearst, 2006a), automatic facet generation (Stoica and Hearst, 2007), and comparing faceted search to the clustering approach of her earlier Scatter/Gather work (Hearst, 2006b).

Around the same time, Gary Marchionini led another effort for faceted search at the University of North Carolina, Chapel Hill, called the Relation Browser (Zhang and Marchionini, 2005; Capra and Marchionini, 2008). The project was originally developed for the US Bureau of Labor Statistics, aiming to improve searching and navigation in the bureau's web site. In additional to faceted navigation, Relation Browser also has certain data analyzing functionalities (*e.g.*, showing histograms of selected data). Marchionini and colleagues have been able to iteratively improve their interface based on several years of user studies (Marchionini and Brunk, 2003; Zhang et al., 2004).

In terms of commercial applications, faceted search is widely used by e-commerce sites now, such as Amazon[7] and eBay[8]. We have shown the (simplified) faceted search interface from Amazon in Figure 2.2. From the example, we can see that faceted search very naturally suits the exploratory product search task – facets in faceted search summarize important properties of products that a customer may be need to consider and provide control for the customer refine the product search results from different perspectives.

There also open-source projects for faceted search. Perhaps the most well-know one is Apache Solr (Smiley and Pugh, 2011). It is an open source search platform supporting faceted search and full-text search. It powers the search functionality of public sites such as CNET, Zappos, Netflix, as well as other government and cooperate intranet sites (Wei et al., 2013).

2.4 Faceted Web Search

The principle of a faceted taxonomy used in faceted search is widely applicable. In research literature, there are faceted search systems developed for a wide range of domains, including images (Yee et al., 2003), movies (Koren et al., 2008), desktop content (Cutrell et al., 2006), audio content (Diao et al., 2010), and houses (Shneiderman, 1994). For commercial applications, faceted search has been adopted by vendors such as Endeca[9] and IBM[10], and (as mentioned before) it is now the dominant search paradigm used in e-commercial sites (*e.g.*, Amazon and eBay)

Despite the success of faceted search in vertical applications, there is limited work in exploring **Faceted Web Search**, an extension of faceted search to the open-

[7]http://www.amazon.com

[8]http:/www.ebay.com

[9]http://en.wikipedia.org/wiki/Endeca

[10]https://en.wikipedia.org/wiki/IBM

domain web (Kong and Allan, 2014). In the open-domain web setting, the corpus or the collection of information objects considered is the entire web, not restricted to any given domains as in the vertical applications. But as with conventional faceted search, Faceted Web Search should also provide facets to assist users in the same principle. We have shown an example of Faceted Web Search in Figure 1.2, in which the user searches with the web query "*baggage allowance*", and the Faceted Web Search system provides a list of facets (independent taxonomies) including as a facet for different airlines, {"*Delta*", "*JetBlue*", "*AA*", ...}, a facet for different flight types, {"*domestic*", "*international*"}, and a facet for different classes, {"*first*", "*business*", concepteconomy}. The user can then select terms in the facets to refine or explore the search results.

The challenge of Faceted Web Search stems from the fact that the web is very large and heterogeneous, as discussed by Teevan et al. (2008). From a system perspective, because the web is very large, it is infeasible to build a complete faceted taxonomy for the entire web manually. And because the web is heterogeneous, it is also difficult to develop automatic methods for generating facets for the entire web. From a user perspective, users conduct a wide range of search tasks on this large and heterogeneous corpus. Queries applied to the web are varied in intent, which makes it difficult to predefine a set of facets for all the queries.

To cope with the challenge, we extract facets for a given query from its search results directly. To differentiate "*facets*" in conventional faceted search (as defined in Definition 2.3.1), we call the facets generated for a particular query "*query facets*". Ideally, a **query facet** should just be a facet that is relevant to the given query. So it should be a hierarchal tree of concepts (represented by their text labels). However, as a start, the query facets we extracted are one-level in this work, which is described below. We leave the extension to two-level (or hierarchal) query facets to future work.

For a **two-level query facet**, there is a parent node and several child nodes. For example, in Figure 2.3, the query facet "*airline*" for query "*baggage allowance*" is two-level. "*Airline*" is the parent node, and "*AA*", "*Delta*", "*JetBlue*" are the child nodes. When there are only two levels in a query facet, we can call the parent node

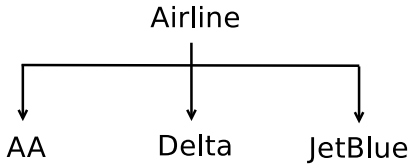


Figure 2.3: A two-level query facet for the query "baggage allowance"

(or more precisely, its presentation term) a **facet label** (*e.g.*, "*airline*"), and the child nodes **facet terms** (*e.g.*, "*AA*", "*Delta*", "*JetBlue*"). Because all the facet terms are directly subsumed by the facet label (by the definition of taxonomy), facet terms are instances of the same semantic class or category (*i.e.*, the semantic class represented by their parent node). From a user's perspective, the facet terms succinctly represent different options in the same category (represented by the facet label) that a user can select to refine the issued query.

Our work generates one-level query facets. A **one-level query facet** is simply a two-level query facet with the facet label omitted (or missing). In other words, a one-level query facet is a set of facet terms (*e.g.*, {"*AA*", "*Delta*", "*JetBlue*"}) without their facet label (*e.g.*, "*airline*"). But as in a two-level query facet, facet terms in a one-level query facet are subsumed by the *implicit* or *missing* facet label. Equivalently, a one-level query facet can be defined as follows:

Definition 2.4.1. A **one-level query facet** is a set of terms that are subsumed by an implicit label. The set of terms represents different options in the same category that a user can select to refine the issued query.

Here the set of "*terms*" corresponds to the "*facet terms*" in a two-level query facet. The "*implicit label*" corresponds to the missing "*facet label*" in the two-level query facet.

2.5 Other Related Techniques

There are a number of techniques developed to achieve similar goals as faceted search or faceted web search. In this section, we discuss query subtopic mining, search result diversification, search result clustering/organization, query suggestion and semantic class extraction.

2.5.1 Query Subtopic Mining

To address multi-faceted queries, much previous work studied mining query subtopics (or aspects). A query subtopic is often defined as a distinct information need relevant to the original query. It can be represented as a set of terms that together describe the distinct information need (Wang et al., 2009; Wu et al., 2011; Dang et al., 2011) or as a single keyword that succinctly describes the topic (Song et al., 2011). For example, {"*news*", "*cnn*", "*latest news*", "*mars curiosity news*"} is a query subtopic for the query *mars landing*, which describes the search intent of Mars landing news. {"*photos*", "*NASA*", "*new photos*", "*curiosity rover photos*"} is another query subtopic for the query, which describes the search intent about Mars landing photos.

Different resources have been used for mining query subtopics. Wang and Zhai (2007) and Hu et al. (2012) used related queries from search logs as candidates, and clustered them into query subtopics. Wang and Zhai (2007), for example, used snippets of a query's clicked web documents to enrich the query representation, and then cluster related past queries into query subtopics. Due to data sparsity for instance-level query subtopics, some work (Wang et al., 2009; Xue and Yin, 2011; Wu et al., 2011; Yin and Shah, 2010) mined generic query subtopics, which are query subtopics

for a generic class of queries. For example, Yin and Shah (2010) built a taxonomy of query subtopics for categories of name entity queries using search logs. Wu et al. (2011) also worked on identifying query aspects for named entities queries. They propagated reformulation phrases for a classes of named entities queries. Other than query logs, query subtopics can also be mined from documents. For example, Dang et al. (2011) worked on clustering related anchor texts in ClueWeb09 corpus into query subtopics. Allan and Raghavan (2002), from a text corpus, extracted commonly occurring parts of speech pattern near a single-word query to find different potential specifications of the query.

Query subtopics and facets are different in that the terms in a query subtopic are not restricted to have any specific semantic relations or structures. However, the terms (more precisely the concepts, defined in Section 2.1.1) in a facet are organized in the taxonomy tree, and they need to subsume its child terms in the tree. For example, the query subtopic {"*news*", "*cnn*", "*latest news*", "*mars curiosity news*"} describes the search intent of Mars landing news, but there are no specific semantic relations between the terms in it, and thus it is not a facet. Instead, a valid facet that describes Mars landing news could be nodes "*cnn*", "*abc*", "*fox*" with a parent node "*news channels*", where the parent node "*new channels*" subsumes all the child nodes by the IS-A relations between them.

2.5.2 Search Result Diversification

Search result diversification has been studied as a method of tackling ambiguous or multi-faceted queries, while a ranked list of documents remains the primary output feature of Web search engine today. The purpose is to diversify the ranked list to account for different search intents or query subtopics. Techniques studied for search result diversification can be classified by whether or not they explicitly represent the

query subtopics in their models. As a result, they are often categorized as being implicit or explicit.

Implicit approaches (Carbonell and Goldstein, 1998; Zhai et al., 2003) try to select documents that are different from the previously selected documents to reduce redundancy or increase novelty, without explicitly modeling the actual subtopics. For example, the pioneer implicit approach is known as Maximal Marginal Relevance (Carbonell and Goldstein, 1998). This technique was originally proposed to reduce redundancy in document rankings as well as in text summarization. It scores each candidate document by its estimated relevance to the search query discounted by its maximum similarity with respect to the documents that have been selected earlier. Then it uses a greedy algorithm to select/rank documents based on the scores, as exactly maximizing the diversification objective is NP-hard.

Explicit approaches (Agrawal et al., 2009; Carterette and Chandar, 2009; Santos et al., 2010; Dang and Croft, 2012, 2013) instead model query subtopics explicitly and then directly select documents that cover different subtopics. For example, Agrawal et al. (2009) used the Open Directory Project taxonomy (a single taxonomy, not a faceted taxonomy) to model query subtopics. Queries are classified into categories (nodes) in the taxonomy, and these categories are used as query subtopics. Then, documents are favored if they are classified into the categories that are less-represented by the documents that have been selected earlier. Note that different from facets in faceted search, these the categories are not explicitly presented to users; the categories are only used as representations of query subtopics for search result diversification. Santos et al. (2010) used a similar diversification model as Agrawal et al. (2009). The difference between them is: Agrawal et al. (2009) used the Open Directory Project taxonomy to represent query subtopics, while Santos et al. (2010) used the query suggestions obtained from commercial search engines for each query as its query subtopics.

Search result diversification could increase the likelihood that users will find documents relevant to their specific search intent. However, a weakness of this technique is that the query subtopics are hidden from the user, leaving him or her to guess at how the results are organized. Faceted Web Search addresses this problem by explicitly presenting different facets of a query for users to select. The presented facets not only provide an overview from different aspects about the search tasks, but also offer control for users to navigate the search results.

2.5.3 Search Result Clustering and Organization

Search result clustering is a technique that organizes search results by grouping them into, usually labeled, clusters by query subtopics (Cutting et al., 1992; Käki, 2005; Zamir and Etzioni, 1999; Carpineto et al., 2009). It not only offers a complementary view to the flat ranked list of search results, but also provide users the ability to choose the clusters of interest in an interactive manner. An extensive survey for search result clustering is provided by Carpineto et al. (2009). In the following, we provide a brief discussion of several prominent search result clustering systems.

Scatter/Gather (Cutting et al., 1992; Hearst and Pedersen, 1996) is an landmark example of search result clustering systems. The system clusters documents into a set of groups, and each group is labeled with a few frequently occurring words in the cluster's documents. Then a user is expected to select one or two clusters which are thought to contain the documents of interest. After that the selected groups are re-clustered and again labeled. The user can continues this process of drilling down until a satisfactory group of documents is gathered.

One major limitation in Scatter/Gather is the generated cluster labels can be difficult for users to interpret. The labels are simply sets of frequently occurring words, and thus they may not be meaningful to the users. Zamir and Etzioni (1999, 1998) addressed the problem by using phrases that appear frequently in the clustered

documents as labels. To support online processing, they developed the Suffix Tree Clustering algorithm for clustering and identifying phrase labels efficiently.

In addition to research projects, search result clustering is also used in commercial search engines. The first commercial application is probably Northern Light in the end of the 1990s. The system is based on a set of predefined categories for web documents, and search results are clustered by their categories. After Northern Light, a major breakthrough was made by Vivisimo, which generates clusters and cluster labels for search results dynamically.

Instead of organizing search results in groups, there is also some work (Lawrie et al., 2001; Lawrie and Croft, 2003; Nevill-Manning et al., 1999) that summarizes search results or a collection of documents in a topic hierarchy or a single taxonomy. For example, previous studies (Lawrie et al., 2001; Lawrie and Croft, 2003) used a probabilistic model for creating topical hierarchies, in which a graph is constructed based on conditional probabilities of words, and the topic words are found by approximately maximizing the predictive power and coverage of the vocabulary.

Search result clustering or organization provides navigation capability over the search results. In this aspect, they are similar to faceted search (or Faceted Web Search); both of them combine direct search with navigational search. However, they are fundamentally different in that: faceted search provides faceted navigation based on a faceted taxonomy (multiple independent taxonomies), while search result cluster or organization provides "directory-like" navigation based on a single taxonomy. As discussed in Section 2.1.2 and Section 2.3, the strict ordering imposed by a single taxonomy is too rigid when dealing with compound information objects in multi-faceted information space. Faceted navigation in faceted search allows users to combine facets to express a complex information need and filter/re-rank search results from multiple aspects. The utility of faceted search interface was investigated in various studies (Pollitt, 1998; Hearst, 2006b; Pratt et al., 1999; Yee et al., 2003; Käki,

2005; Rodden et al., 2001), where it was shown that users engaged in exploratory tasks often prefer the faceted search interface over simple ranked result list, as well as the alternative ways of organizing search results.

2.5.4 Query Suggestion

Query suggestion (or query recommendation) is a common technique used by search engines to assist users in reformulating queries. In the suggestion process, a user starts with issuing an initial query that may be ineffective. Then, the system provides a set of alternative queries that may better address the user's information need as suggestions. After that, the user can select one of the query suggestion to search again. We show an example in Figure 2.4, in which a list of related queries for the query "*baggage allowance*" are suggested.

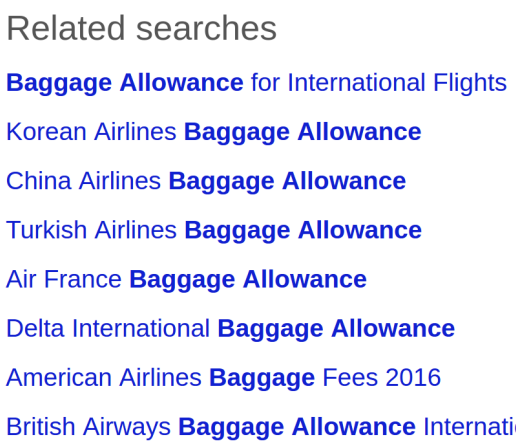


Figure 2.4: Example query suggestions for the query "*baggage allowance*"

There has been significant previous work on query suggestion. We briefly review some of it below. Most of the previous work relies on query logs for query suggestion (Baeza-Yates et al., 2004; Jones et al., 2006; Mei et al., 2008; Ozertem et al., 2011). For example, Baeza-Yates et al. (2004) provided query suggestions by clustering related queries in query logs, and Jones et al. (2006) suggested strongly related queries identified from pairs of successively issued queries found in query logs. One

widely used technique is exploiting query-click graphs (Craswell and Szummer, 2007; Mei et al., 2008; Ozertem et al., 2011). The query-click graph (Craswell and Szummer, 2007) is a bipartite graph consists of nodes representing queries and documents. The two types of nodes are connecting by edges representing clicks on the document for that query. By performing a random walk on this bipartite graph, query similarity can be calculated, and more similar queries can be shown as suggestions.

In the cases where query logs are not available, only a few methods have been proposed for query suggestions. Bhatia et al. (2011) relied on the corpus to find related queries as suggestions. They extracted frequently occurring phrases and n-grams from the text corpus, and used them as suggestions for auto-completing the query that a user is typing. Luo et al. (2008) relied on MeSH (Medical Subject Headings), a medical thesaurus, to find related medical phrases as query suggestions for a medical search engine.

Query suggestion and Faceted Web Search are similar in that both of them provide terms for users to select and reformulate queries. However, they are very different in the philosophy behind. In query suggestion, the system provides related queries or queries similar to the initial issued query, and the provided suggestions are used to replace the initial query. In Faceted Web Search, the system provides facets for the query, which describe the search intent from different aspects and are used to refine the initial query instead of replacing it. Due to the difference in the philosophy behind, the way they present suggestions/facet terms are very different. Query suggestions are presented (usually) as a flat list, as shown in Figure 2.4. Faceted Web Search instead presents facet terms in a more structured way. The facet terms are grouped together in query facets. For example, in Figure 1.2, "*AA*", "*Delta*", "*JetBlue*" are grouped together under conceptAirline. By grouping terms into query facets, the facet interface essentially provides a skip list of these facet terms for users. More specifically, a user can skip an entire query facet if the user finds the facet irrelevant,

while in the query suggestion interface, the user needs to examine each suggestions one by one.

2.5.5 Semantic Class Extraction

Semantic class extraction is to automatically mine semantic classes represented as their class instances from certain data corpora. For example, it may extract *USA*, *UK*, *China* as class instances of semantic class *country*. Due to the similar semantic relationships between terms inside a facet and a semantic class, semantic class extraction can be used for facet generation.

Existing approaches can be roughly divided into two categories: distributional similarity and pattern-based (Shi et al., 2010). The distributional similarity approach is based on the distributional hypothesis (Harris, 1954), that terms occurring in analogous contexts tend to be similar. Different types of contexts have been studied for this problem, including syntactic context (Lin, 1998; Pantel and Lin, 2002) and lexical context (Pantel et al., 2004; Agirre et al., 2009; Pantel et al., 2009). Lin (1998) constructed syntactic context using dependency triples extracted from text corpus. A dependency triple consist of two words and the grammatical relationship between them in the input sentences. The dependency triples are aggregated as syntactic context for each words. The construction of syntactic contexts requires sentences to be parsed by a dependency parser, which may be extremely time-consuming on large corpora. As an alternative, lexical context can be constructed more efficiently. For example, Agirre et al. (2009) simply constructed lexical context by using the the surrounding words in sentences. Based on the context representation, clustering algorithms can be applied to cluster similar words/phrases together as a semantic class (Pantel and Lin, 2002).

The pattern-based approach applies textual patterns (Hearst, 1992; Pasca, 2004), HTML patterns (Shinzato and Torisawa, 2005) or both (Zhang et al., 2009; Shi et al.,

2010) to extract instances of a semantic class from some corpus. For example, Hearst (1992) used pattern "*NP such as NP, NP, and NP*" to extract hyponyms and hypernyms, where the hyponyms can be used as candidates for semantic class instances. Shinzato and Torisawa (2005) used patterns to extract HTML itemized-lits as semantic class. Our work uses both types of extraction patterns for facet generation, and they are discussed in details in Section 3.4. The raw semantic class extracted can be noisy. To address this problem, Zhang et al. (2009) used topic modeling to refine the extracted semantic classes. Their assumption is that, like documents in the conventional setting, raw semantic classes are generated by a mixture of hidden semantic classes.

In this work, we apply pattern-based semantic class extraction on the top search results to extract candidates for facet generation (Section 3.4), and design some features based on distributional similarity for refining facet candidates (Section 3.6.2.4).

2.6 Summary

Faceted search is the combination of directed search and faceted navigation which enables users to search and navigate through a multi-dimensional information space. Faceted navigation is based on a faceted taxonomy consists of a set of independent taxonomies (called facets) to be combined for expressing compound information needs. Each of the independent taxonomies or facets is a tree of concepts with all concepts subsuming their descendant concepts. A concept in a taxonomy is an abstraction which identifies all the information objects classified under it, and it is often presented as a term to convey its meaning to users.

Faceted Web Search, the focus of this work, is the extension of faceted search to the open-domain web setting. In the open-domain web setting, the corpus or the collection of information objects considered is the entire web, not restricted to any given domains. But as with conventional faceted search, Faceted Web Search should

also provide facets to assist users in the same principle. For Faceted Web Search, we directly extract facets for queries from search results, called "*query facets*". A query facet is a facet that is relevant to the given query, and ideally the facet can be a two-level or multiple-level tree. A two-level query facet consist of a parent node (called facet label) and its child nodes (called facet terms). The facet terms are directly subsumed by their facet label (by the definition of taxonomy). However, as a start for Faceted Web Search, this work only studies one-level query facets. A one-level query facet is simply a two-level query facet without an explicit facet label. In other words, a one-level query facet is a set of terms (facet terms) that are subsumed by their missing or implicit facet label. The set of terms represents different options in the same category that a user can select to refine the issued query. In the rest of this thesis, for the sake of convenience, we use "*query facets*" to refer one-level query facets.

Faceted Web Search as we propose in this work is different from all the past work. It extends conventional faceted search from a fixed-domain setting to an open-domain web setting. It is different from search result diversification in that instead of hiding those query subtopics from users, it explicitly presents different facets of a query. It is different from search results clustering or organization in that instead of directly organizing the search results into a single taxonomy (or different groups), Faceted Web Search provides multiple facets to support faceted navigation. It is also different from query suggestion in that query facets present search refinement suggestions (facet terms) in a more structured way; by grouping facet terms into query facets, the Faceted Web Search interface essentially provides a skip list of these facet terms for users.

We study three main issues of Faceted Web Search that have not been explored in previous work, including facet generation, facet feedback and evaluation for Faceted Web Search. Facet generation for Faceted Web Search is different from query subtopic

mining because of the different nature of query subtopics and query facets. It is also different from semantic class extraction in that it targets a general web query instead of a semantic class. Facet feedback for Faceted Web Search is different from other user feedback because of their different purposes (discussed in Section 6.2). Our evaluation for Faceted Web Search is also different from previous ones for faceted search in that we do not rely on expensive user studies, and thus our evaluating methods are relative cheap to extend for evaluating new systems.

CHAPTER 3

QUERY FACET EXTRACTION

3.1 Introduction

In conventional faceted search, facets are generated in advance for an entire corpus (Stoica and Hearst, 2007; Dakka and Ipeirotis, 2008) either manually or semi-automatically, and then recommended for particular queries (Teevan et al., 2008). However, this approach is difficult to extend to faceted web search due to the large and heterogeneous nature of the web: because the web is very large, it is difficult to assign quality facets to every document in the collection and to retrieve the full set of search results and their associated facets at query time; and because the web is heterogeneous, it is difficult to apply the same facets to every search result or every query.

To cope with this challenge, in this chapter, we propose an alternative solution, called **query facet extraction** (Kong and Allan, 2013), which extracts facets for queries (called query facets) from their web search results. For example, when users search with the query "*baggage allowance*", the system might extract query facets like airlines, {"*Delta*", "*JetBlue*", "*AA*", ...}, travel classes, {"*first*", "*business*", "*economy*"}, and flight types, {"*international*", "*domestic*"}. Changing from a global approach that generates facets in advance for an entire corpus to a query-based approach that extract facets from the top-ranked search results, query facet extraction appears to be a promising direction for solving the open-domain faceted search problem – it not only makes the facet generation problem easier, but also addresses the facet recommendation problem at the same time.

Note that the query facets we extracted are one-level (Section 2.3.1). A one-level **query facet** is a set of terms (*e.g.*, {"*AA*", "*Delta*", "*JetBlue*",...}) that are subsumed by an implicit label (*e.g.*, "*airlines*"). The set of terms succinctly represents different options in the same category (*e.g.*, "*airlines*") that a user can select to refine the issued query (*e.g.*, "*baggage allowance*"). We leave the extension to two-level or hierarchical query facets to future work. As mentioned before, for the sake of convenience, we use "*query facets*" to refer one-level query facets in the rest of this thesis, unless otherwise specified. In Table 3.1, we show (one-level) query facets for three example queries. We will using the first query "*mars landing*" as an example for explanation. The first query facet, {*Curiosity*, *Opportunity*, *Spirit*}, includes different Mars rovers. The second query facet, {*USA*, *UK*, *Soviet Union*}, includes countries relevant to Mars landings. The last facet, {*video*, *pictures*, *news*}, includes different types of media. We can see that the terms in each facet are instances of the same semantic class, and they are all subsumed by their class labels (*e.g.*, "*Mars rovers*", "*countries*", "*media*").

Table 3.1: Query facet examples for three queries

Query 1: Mars landing
Query Facet 1: Curiosity, Opportunity, Spirit Query Facet 2: USA, UK, Soviet Union Query Facet 3: video, pictures, news
Query 2: baggage allowance
Query Facet 1: AA, Delta, Jetblue, ... Query Facet 2: international, domestic Query Facet 3: first class, business class, economy class Query Facet 4: weight, size, quantity
Query 3: Mr Bean
Query Facet 1: comics, movies, tv, books Query Facet 2: The Curse of Mr Bean, Mr Bean Goes to Town, ... Query Facet 3: Rowan Atkinson, Richard Wilson, Jean Rochefort, ... Query Facet 4: Mr Bean, Irma gobb, Rupert, Hubert, ...

In this work, we use query facet extraction to address the problem of facet generation in Faceted Web Search. This approach first extracts candidate facets from the top search results based on textual and HTML patterns, and then refines the extracted candidates, which are often very noisy, using clustering methods. We develop a supervised method based on a graphical model for refining facet candidates. The graphical model learns how likely it is that a term should be selected from the candidates and how likely it is that two terms should be grouped together in a query facet. Further, the model captures the dependencies between the two factors. We propose two algorithms for approximate inference on the graphical model since exact inference is intractable. This proposed method can easily incorporate a rich set of features and learn from available human labels.

The rest of this chapter is organized as follows. We first define the task of query facet extraction and related concepts in Section 3.2, and then describe a general framework to solve this problem in Section 3.3. We propose a supervised clustering method based on a graphical model for refining extracted candidates in Section 3.6. Last, we describe other methods that can be used for refining extracted candidates, including topic modeling (e.g., pLSA, LDA) and a variation of quality threshold clustering model (Dou et al., 2011) in Section 3.7.

3.2 Task Description

3.2.1 Query Facet

A **query facet** is a set of terms (*e.g.*, {"*AA*", "*Delta*", "*JetBlue*",...}) that are subsumed by an implicit label (*e.g.*, "*airlines*"). The set of terms succinctly represents different options in the same category (*e.g.*, "*airlines*") that a user can select to refine the issued query (*e.g.*, "*baggage allowance*"). We call the terms inside a query facet **facet terms**, which can be single words (e.g. "*international*", "*domestic*" in Table 3.1) or phrases (e.g. "*first class*", "*business class*" in Table 3.1).

According to the definition, a query facet have two properties. First, the facet terms inside a query facet should be coordinate terms. Coordinate terms are terms that share a semantic relationship by being subsumed by a more general hypernym. This property is easy to see according to our definition for a query facet. Second, a query facet should be relevant to the query. Otherwise, the query facet is useless for the search task.

Again, note that this definition of query facets corresponds to a one-level faceted taxonomy, in which only information objects that belong to a same parent node are shown as a query facet (Section 2.4). We leave generating query facets as two or more level taxonomies to future work.

When it is clear from context, we will simply use "facet" for "query facet", and "term" for "facet term" for convenience.

3.2.2 Query Facet Extraction

Based on the definitions above, query facet extraction is to extract query facets for a given query from certain resources. While a variety of different resources can be used for query facet extraction, such as a query log, anchor text, taxonomy and social folksonomy, in this work, we only focus on extracting query facets from the top ranked web search results, and and leave others as future work.

3.3 Solution Framework

The idea for solving query facet extraction is to leverage coordinate terms found in the web search results to build high-quality query facets. These coordinate terms can be found by looking into the list structures in webpages (*e.g.*, ordered lists, drop-down lists) or analyzing the linguistic list structures in the textual content (*e.g.*, "*The airlines servicing this airport are AA, Delta, and JetBlue*"). Previous work in

semantic class extraction (Hearst, 1992; Pasca, 2004; Kozareva et al., 2008; Shi et al., 2010) has studied patterns for extracting these structures.

This idea is based on the following two assumptions, which are related to the two properties of query facets described above:

(1) The list structures consist of coordinate terms (terms that are subsumed by the same hypernym). The coordinate terms share ppeer relationship. According to webpage design conversions, webpage editors often list peering information objects in the HTML list structures. Similarly, the linguistic list structures are often used to list peer information objects in writing.

(2) The list structures present relevant and important aspects of the query. Assuming the search results are relevant to the query, those list structures extracted from the search result should also be related to the query. And, if they occur frequently, they may also be important to that query.

Based on the idea, we develop the following general solution framework for query facet extraction, as also illustrated in Figure 3.1:

(1) **Retrieval**: in the first step, given a query, we retrieve the top search results.

(2) **Candidate extraction**: in the second step, we extract list structures as facet candidates from the search results based on pre-defined extraction patterns.

(3) **Facet Refining**: the facet candidates extracted are often very noisy, and cannot be directly used as query facets. In the last step, we refine the candidates to final query facets by re-clustering facets or terms in the candidate set.

We will describe candidate extraction and facet refining in more detail in the next two sections.

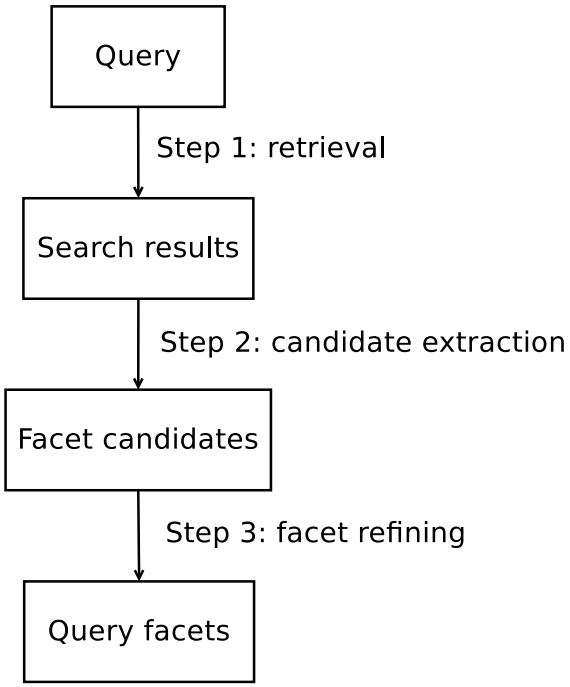


Figure 3.1: Query facet extraction framework

3.4 Candidate Extraction

Following Dou et al. (2011), we use a pattern-based semantic class extraction approach (Shi et al., 2010) to extract lists of coordinate terms from search results as facet candidates. In pattern-based semantic class extraction, instances of a semantic class (*e.g.*, instance "*AA*", "*Delta*", "*JetBlue*" for class "*airlines*") are extracted from textual or webpage corpus based on lexical patterns (Hearst, 1992; Pasca, 2004), HTML patterns (Shinzato and Torisawa, 2005), or both (Shi et al., 2008; Zhang et al., 2009). For example, the pattern "*NP*, *NP*, ..., and *NP*", where "*NP*" stands for a noun phrase, can be used to extract coordinate terms (as instances) from text. Besides lexical patterns, HTML patterns are often used on HTML documents to extract coordinate terms from some HTML structures, like unordered lists (*i.e.*, <UL>), drop-down lists (*i.e.*, <SELECT>) and tables (*i.e.*, <TABLE>). The coordinate terms extracted from each patterns form candidates for query facets, which we call **candidate list**.

We apply both of the two types of patterns on the search results to extract candidate lists. These extraction patterns are summarized in Table 3.2. We describe them in detail in the following two sections.

Table 3.2: Candidate list extraction patterns. All matched *items* in each pattern are extracted as a candidate list.

Type	Pattern
Lexical	*item* {,*item*}* {,} (and|or) {other} *item*
HTML	<select><option>*item*</option>...</select>
	item ...
	item ...
	<table><tr><td>*item*<td>...</table>

3.4.1 Lexical Patterns

We use the following lexical pattern:

item {,*item*}* {,} (and|or) {other} *item*

We apply the pattern on the textual content (ignoring HTML tags and formatting in the webpage) of the search results, and extract matched *items* as a candidate list. To give an example, for the sentence, "*The airlines servicing this airport are AA, Delta, and JetBlue* ", we can extract the candidate list {"*AA*", "*Delta*", "*JetBlue*"} using the lexical pattern. For this lexical pattern, we also restrict those *items* to be siblings in the parse tree of that sentence in order to improve extraction quality. This is because siblings are likely to be coordinate terms; they may be subsumed by or have same semantic relationships to their parent node in the parse tree. We use the PCFG parser (Klein and Manning, 2003) implemented in Stanford CoreNLP (Manning et al., 2014) for parsing documents.

3.4.2 HTML Patterns

We also extract candidate lists based on several HTML patterns that target list structures in HTML webpages, including drop-down lists, ordered lists, unordered

lists and tables. Table 3.3 shows some HTML code examples for these list structures. These HTML list structures are extracted based on the HTML patterns listed in

Table 3.3: HTML code examples for drop-down lists (SELECT), ordered lists (OL), unordered lists (UL) and tables (TABLE).

SELECT
<select> <option value="1">first class</option> <option value="2">business class</option> <option value="3">economy class</option> </select>
OL
  - checked baggage allowance   - carry on baggage allowance   - excess baggage allowance
UL
   Courtesy bags   Dangerous goods   Electronic devices   Sports equipment
TABLE
<table> <tr><td></td><td>economy</td><td>business</td>first</tr> <tr><td>domestic</td>2 bags</td><td>3 bags</td><td>4 bags</td></tr> <tr><td>international</td>1 bag</td><td>2 bags</td><td>3 bags</td></t> </table>

Table 3.2. Note that we do not match the HTML patterns with the HTML code exactly. Instead, we parse the HTML page into objects and extract textual content in the tags that we are interested in. We describe the extraction in details below:

- **SELECT**: For the SELECT tag, we extract textual content in the OPTION tags as a candidate lists. For the example in Table 3.3, we will extract candidate list {"*first class*", "*business class*", "*economy class*"}.

- **OL**: For the OL tag, we extract textual content in the LI tags as a candidate lists. For the example in Table 3.3, we will extract candidate list {"*checked baggage allowance*", "*carry on baggage allowance*", "*excess baggage allowance*"}.

- **UL**: Similarly, for the UL tag, we also extract textual content in the LI tags as a candidate lists. For the example in Table 3.3, we will extract candidate list {"*Courtesy bags*", "*Dangerous goods*", "*Electronic devices*", "*Sports equipment*"}. Note that in this case, when extracting textual content in the LI tags, we ignore other HTML tags/formatting (*i.e.*, "<*a*>").

- **TABLE**: for HTML tables, following Dou et al. (2011), we extract candidate lists from each columns and each rows. For the example in Table 3.3, we will extract 7 candidate lists. To list a few, {"*economy*", "*business*", "*first*"} and {"*domestic*", "*2 bag*", "*3 bags*", "*4 bags*"} are extracted from the first two rows. {conceptdomestic, "*international*"} and {"*economy*", "*2 bags*", "*1 bag*"} are extracted from the first two columns.

Ordered lists (OL) and unordered lists (UL) can be nested, as shown in Figure 3.2. For nested HTML lists, we extract all sibling items from each level. For the example in Figure 3.2, we will extract two lists, {"*Coffee*", "*Tea*", "*Milk*"} and {"*Black tea*", "*Green tea*"}.

3.4.3 Candidate List Cleaning

After extracting candidate lists from the top ranked search results, we further clean them as follows:

(1) First, all the list items are normalized by converting text to lowercase and removing non-alphanumeric characters.

(2) Then, we remove stopwords and duplicate items in each list.

```
<ul>
 <li>Coffee</li>
 <li>Tea
  <ul>
   <li>Black tea</li>
   <li>Green tea</li>
  </ul>
 </li>
 <li>Milk</li>
</ul>
```

Figure 3.2: An example for nested HTML lists.

(3) Finally, we discard all lists that contain only one item or more than 200 items.

After the cleaning process, we harvest a set of candidate lists, from which we want to build high-quality query facets.

3.5 Facet Refining

The candidate lists extracted are usually noisy (Zhang et al., 2009), and could be non-relevant to the issued query, therefore they cannot be used directly as query facets. For example, Table 3.4 shows four candidate lists extracted for the query "*baggage allowance*". L_1 contains terms that are relevant to "*baggage allowance*", but they are not coordinate terms – "*delta*", "*france*" and "*round-trip*" are not members of the same class. L_2 is a valid query facet, but it is incomplete – another airline "*aa*" appears in L_3. L_3 is mixed with different facets, airlines and travel classes. L_4 is non-relevant to the query.

Table 3.4: Four candidate lists for the query "*baggage allowance*"

L_1: delta, france, round-trip
L_2: delta, jetblue, british airways
L_3: aa, first, business, economy
L_4: hard to remember, playing it by ear, ...

Since the candidate lists are frequently noisy, we need an effective way to refine extracted candidate lists into high-quality query facets. We call this problem **facet refining**, in which we take a set of candidate lists as input, and want to output high-quality query facets. This facet refining problem is the core issue in query facet extraction, and a main focus of this chapter. Existing query facet extraction models differ in how they refine candidate lists.

One related work about facet refining (Dou et al., 2011) clusters similarity candidate lists together as query facets (called query dimensions in their original paper), and then ranks/selects clusters and cluster items based on heuristics scores. We find this method is difficult to incorporate features into. It also does not have the flexibility of breaking a candidate list into two query facets. We describe more about this method and other related method for facet refining in Section 3.7. We also compare these methods with our models in Chapter 4 and Chapter 7.

We instead treat the facet refining problem as a **selective clustering** problem. In the selective clustering problem, we do not cluster all given items, but only cluster a subset of the items. In the case of facet refining, we want to discard noisy terms, and cluster only facet terms in the candidate lists (*e.g.*, "*aa*", "*delta*", "*jetblue*", "*british airways*", "*first*", "*business*" and "*economy*" in Table 3.4) into query facets (*e.g.*, {"*aa*", "*delta*", "*jetblue*", "*british airways*"} and {"*first*", "*business*","*economy*"}). We present this problem more formally in Section 3.6.1.

To address this problem, we develop a supervised method based on a graphical model, presented in the next section.

3.6 Query Faceting Models

In this section, we describe query faceting (QF) models, our supervised methods based on a directed graphical model for facet refining. A directed graphical model (or Bayesian network) is a graphical model that compactly represents a probability

distribution over a set of variables (Pearl, 1988). It consists of two parts: 1) a directed acyclic graph in which each vertex represents a variable, and 2) a set of conditional probability distributions that describe the conditional probabilities of each vertex given its parents in the graph.

We treat facet refining as a selective clustering problem (as described in Section 3.5), and solve it as a labeling problem, in which we are trying to predict 1) whether a list item is a facet term, and 2) whether a pair of list items is in a same query facet. Then, we used a directed graphical model to exploit the dependences that exist between those labels. Similar to conditional random fields (Lafferty et al., 2001), we directly model the conditional probability $P(y|x)$, where y is the label we are trying to predict and x is the observed data – list items and item pairs. Thus, it avoids modeling the dependencies among the input variables x, and can handle a rich set of features. For our graph model, exact maximum a posteriori inference is intractable; therefore, we approximate the results using two algorithms.

In the rest of this section, we first describe the facet refining problem more formally in Section 3.6.1, and then present our graphical model in Section 3.6.2. We describe how to train and perform inference on the model in Section 3.6.3 and Section 3.6.4 respectively.

3.6.1 Problem Formulation

Before diving into the QF method, we first define the facet refining problem more formally. We use $F = \{t_i\}$ to denote a query facet, consisting of a set of facet terms t_i. $\mathcal{F} = \{F_i\}$ denotes the set of query facets for the given query. $T_{\mathcal{F}} = \bigcup_i F_i$ denotes all the facet terms in $\mathcal{F}$. Candidate lists (or candidate facets) are just an imperfect version of query facets, and we substitute "F" with "L" to denote corresponding variables. $L = \{t_i\}$ denotes a candidate list. $\mathcal{L} = \{L_i\}$ denotes all the candidate lists extracted for the query. $T_{\mathcal{L}} = \bigcup_i L_i$ denotes all list items (or terms)

in the candidate lists. Based on the formulation, the facet refining problem is simply to find $\mathcal{F}$ constrained with $T_{\mathcal{F}} \subseteq T_{\mathcal{L}}$, given $\mathcal{L}$ (and possibly other resources).

In our query faceting models, the facet refining problem was treated as a label prediction problem. It aims to learn and predict jointly 1) whether a list item is a facet term and 2) whether a pair of list items are in the same query facet. We denote the two types of labels as follows. The term/item labels are denoted by $Y = \{y_i\}$, where $y_i = 1\{t_i \in T_{\mathcal{F}}\}$ is a label indicating whether a list item t_i is indeed a facet term. Here $1\{\cdot\}$ is the indicator function which takes on a value of 1 if its argument is true, and 0 otherwise. The pair labels are denoted by $Z = \{z_{i,j}\}$, where $z_{i,j} = 1\{\exists\, F \in \mathcal{F},\, t_i \in F \wedge t_j \in F\}$ is a label indicates whether list item t_i and t_j are in the same query facet. Thus, the facet refining problem is now formulated as the problem of predicting labels Y and Z.

3.6.2 The Graphical Model

Our supervised method is based on a directed graphical model, aiming to capture the dependencies between the term and pair labels. The graphical model is shown in Figure 3.3. We further describe it as follows.

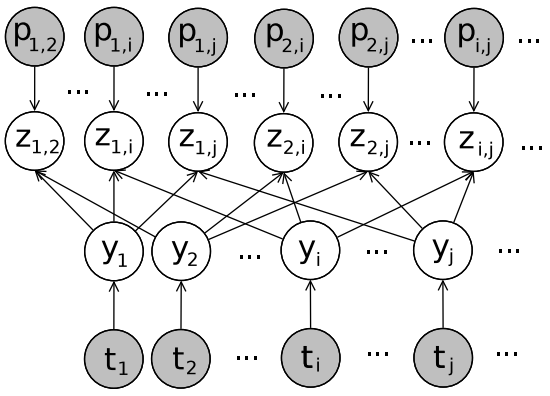


Figure 3.3: A graphical model for candidate list data

3.6.2.1 The Graph

First, we describes the four types of variables in the graphical model as follows. We use the formulation described in Section 3.6.1.

- list items: $t_i \in T_{\mathcal{L}}$, as defined before, are all the list items from the extracted candidate lists.
- item pairs: $p_{i,j} = (t_i, t_j)$ is simply a short name for term pair t_i and t_j. $P_{\mathcal{L}} = \{p_{i,j} | \, t_i, t_j \in T_{\mathcal{L}}, \, t_i \neq t_j\}$ are all the item pairs in $T_{\mathcal{L}}$.
- item labels: $y_i \in Y$, as defined before, are all item labels.
- pair labels: $z_{i,j} \in Z$, as defined before, are pair labels.

List items t_i and item pairs $p_{i,j}$ will be characterized by corresponding features (described in Section 3.6.2.4). They are always observed. Item labels y_i and pair labels $z_{i,j}$ are what we are trying to predict. In summary, the vertices in our graphical model are $V = T_{\mathcal{L}} \cup P_{\mathcal{L}} \cup Y \cup Z$.

Second, as shown in Figure 3.3, there are three types of edges in the graph:

- edges from each list item t_i to its corresponding label y_i.
- edges that point to each item pair label $z_{i,j}$ from the two corresponding list items y_i and y_j.
- edges from each item pair $p_{i,j}$ to its corresponding label $z_{i,j}$.

3.6.2.2 Conditional Probability Distributions

We use logistic-based conditional probability distributions (CPDs) for variable y_i and $z_{i,j}$, because they are simple and can easily incorporate a rich set of features. The CPDs are defined as in Equation 3.1 and Equation 3.2,

$$P(y_i = 1 | t_i) = \frac{1}{1 + \exp\{-\sum_k \lambda_k f_k(t_i)\}}, \tag{3.1}$$

$$P(z_{i,j} = 1|p_{i,j}, y_i, y_j) = \frac{y_i y_j}{1 + \exp\{-\sum_k \mu_k g_k(p_{i,j})\}}, \tag{3.2}$$

where f_k and g_k are features that characterize a list item and a item pair respectively. λ and μ are the weights associated with f_k and g_k respectively. Compared to a conventional logistic function, Equation 3.2 has an extra term, $y_i y_j$, in the numerator. When $y_i = 0$ or $y_j = 0$, we have $P(z_{i,j} = 1|p_{i,j}, y_i, y_j) = 0$. This means when either of the two list items is not a facet term, the two items can never appear in a query facet together. When both of the t_i and t_j are facet terms, $P(z_{i,j} = 1|p_{i,j}, y_i, y_j)$ becomes a conventional logistic function, which models the probability of t_i and t_j being grouped together in a query facet, given the condition that both t_i and t_j are facet term.

3.6.2.3 Joint Conditional Probability

Similar as in conditional random fields (Lafferty et al., 2001), we directly model the joint conditional probability $P(Y, Z|T_{\mathcal{L}}, P_{\mathcal{L}})$. Thus, it avoids modeling the dependencies among the input variables $T_{\mathcal{L}}, P_{\mathcal{L}}$, and can handle a rich set of features. The joint conditional probability for the graphical model is calculated as

$$P(Y, Z|T_{\mathcal{L}}, P_{\mathcal{L}}) = \prod_i P(y_i|t_i) \prod_{i,j} P(z_{i,j}|p_{i,j}, y_i, y_j), \tag{3.3}$$

where the $P(y_i|t_i)$ and $P(z_{i,j}|p_{i,j}, y_i, y_j)$ are defined in Equation 3.1 and Equation 3.2 respectively.

3.6.2.4 Features

There are two types of features used in our graphical model, term features and (term) pair features.

Item features, $f_k(t)$ in the graphical model, characterize a single list item in terms of whether the item is a facet term. There are two factors we consider: (1) relevance to the query and (2) quality as a coordinate term. We design a rich set of features to capture the two factors from different perspectives. More specifically, the

features we designed are based on different data sources, and based on different types of frequency counts as described in details below.

The different data sources we used include a large web corpus (ClueWeb09), the list item itself, and the top search result webpages. For the top search results, we consider extraction on the following fields/parts:

- Content: the textual content of the search result webpages
- Title: the title of the search result webpages
- List: the candidate lists extracted from the search results. We also consider the candidate lists extracted by each extraction pattern (Section 3.4) separately, as we find these patterns are of different extraction qualities (Section 4.4.6).
  - Text: candidate lists extracted based on the lexical pattern
  - Ol: candidate lists extracted based on the OL pattern
  - Ul: candidate lists extracted based on the UL pattern
  - Select: candidate lists extracted based on the SELECT pattern
  - Tr: candidate lists extracted based on the TABLE pattern, and extracted by the rows.
  - Td: candidate lists extracted based on the TABLE pattern, and extracted by the columns.

We extract the following types of frequency counts (on the different fields) as features. Note that these frequency-based features are normalized using $log(frequency+1)$.

- Termfreq: item/term frequency, the frequency of the list item.
- PageFreq: page frequency, the number of search result webpages that contain the list item.

- WpageFreq: weighed PageFreq. Each search result webpage count is weighted by its rank, $1/\sqrt{rank}$. This assume that the webpages in the top ranks are more important.

- SiteFreq: site frequency, the number of unique websites of the search results that contain the list item. SiteFreq addresses the following problem of PageFreq. Some websites have a fixed template for all of its webpages. Search results may contains multiple webpages from the same website. In that case, the candidate lists extracted from the template part may repeat multiple times, and be favored by PageFreq unreasonably.

We list all item features in Table 3.5. To capture the relevance of list item (or term) to the query, we use some TF/IDF-based features extracted based on the search result content (Content), title (Title), a webpage corpus (Global corpus), and their combination (Combination). For example, *ContentTermFreq* is the frequency count of the item in the text content of the top k search results. *TitleSiteFreq* is the number of webpages in the top k search results that contain the item in their title. *IDF* is the inverse document frequency of the list item based on a large web page corpus, ClueWeb09[1]. IDF is useful in down-weighting terms that occur frequently in generally, and thus likely to be less important. We calculate IDF as $IDF(t) = \log \frac{N-N_t+0.5}{N_t+0.5}$, where N is the total number of documents in the collection, N_t is the number of documents that contain t (document frequency). We also combine different features together. For example, *ContentTermFreq.IDF* multiples *ContentTermFreq* with *IDF* (like TF-IDF). *ListTermFreq.ListIDF* multiples *ListTermFreq* with *ListIDF*, where *ListTermFreq* is frequency count of the list item in the candidate lists extracted based on all the patterns. Besides relevance to the query and quality as coordinate terms,

[1]http://lemurproject.org/clueweb09

Table 3.5: Item features

Field/Source		Feature	Description
Content		ContentTermFreq	TermFreq in Content
		ContentPageFreq	PageFreq in Content
		ContentWpageFreq	WpageFreq in Content
		ContentSiteFreq	SiteFreq in Content
Title		TitleTermFreq	TermFreq in Title
		TitlePageFreq	PageFreq in Title
		TitleSiteFreq	SiteFreq in Title
List	ListText	ListTextTermFreq	TermFreq in ListText
		ListTextPageFreq	PageFreq in ListText
		ListTextSiteFreq	SiteFreq in ListText
	ListOl	ListOlTermFreq	TermFreq in ListOl
		ListOlPageFreq	PageFreq in ListOl
		ListOLSiteFre	SiteFreq in List
	ListUl	ListUlTermFreq	TermFreq in ListUl
		ListUlPageFreq	PageFreq in ListUl
		ListUlSiteFreq	SiteFreq in ListUl
	ListSelect	ListSelectTermFreq	TermFreq in ListSelect
		ListSelectPageFreq	PageFreq in ListSelect
		ListSelectSiteFreq	SiteFreq in ListSelect
	ListTr	ListTrTermFreq	TermFreq in ListTr
		ListTrPageFreq	PageFreq in ListTr
		ListTrSiteFreq	SiteFreq in ListTr
	ListTd	ListTdTermFreq	TermFreq in ListTd
		ListTdPageFreq	PageFreq in ListTd
		ListTdSiteFreq	SiteFreq in ListTd
Term		Length	number of words in the term
Global corpus		IDF	inverse document frequency
		ListIDF	IDF in a candidate list collection
Combination		ContentTermFreq.IDF	ContentTermFreq × IDF
		ListTermFreq.ListIDF	ListTermFreq × ListIDF

a facet term should also be succinct. Thus we also use feature *Length*, which counts the number of words in the list item.

To capture how likely item t is to be a coordinate term (or an instance of a semantic class), we use features extracted from candidate lists (List) based on different patterns (ListText, ListOl, ListUl, ListSelect, ListTr, ListTd). For example, *listSelectTermFreq* is the frequency of the list item in the candidate lists extracted based on

the SELECT pattern. *listTextPageFreq* is the number of the search result webpages that contains the list item in candidate lists extracted based on the lexical pattern from the webpages. Some list items occur frequently in candidate lists across different queries, such as *home*, *contact us* and *privacy policy*. They are treated as stopwords, and removed from the candidate lists. We also use *listIDF* to cope with this problem, in a similar way as IDF. *listIDF* is the IDF of a list item in a collection of candidate lists we extracted (see Section 4.2). It is calculated in the same form as *IDF*, as $listIDF(t) = \log \frac{NL - NL_t + 0.5}{NL_t + 0.5}$, where NL is the total number of candidate lists in the collection, NL_t is the number of lists contain the list item t.

Item Pair Features, $g(p_{i,j})$ in the graphical model, are used to capture how likely it is that a pair of list items should be grouped into a query facet, given that the two list item both are facet terms. We design item pair features based on different types of similarity between the two list items, listed in Table 3.6. One straightforward feature is the frequency count of the two list items occurring in a same candidate list (*ListCooccur*). Another one is the difference in the length of the list item (*LengthDiff*), which assumes that similar list item should be of similar length.

Table 3.6: Item pair features. t_i and t_j are an item pair.

Feature	Description
LengthDiff	Length difference in words, $\|Length(t_i) - Length(t_j)\|$
ListCooccur	Number of candidate lists in which t_i, t_j co-occur
TextContextSim	Similarity between text contexts
ListContextSim	Similarity between list contexts

Besides these straightforward features, item similarity can also be measured by their context similarity. One common context for a term is the text content surrounding the term (Shi et al., 2010). The feature based on text content similarity is called *textContextSim* in Table 3.6. We build the text context using surrounding words (within 25 words) of the list item in the search result webpages, and represent the text context as a vector of term frequency weights, then we use cosine similarity to

calculate *TextContextSim*. Another context we propose in this work is based on the candidate lists extracted, which we call list context. We use list items that appears together with the given list item in the same candidate lists as the list context. An example is given in Table 3.7. In the example, the list context for "*Delta*" include list item "*AA*" (twice), "*JetBlue*" (twice), "*Southwest*", and the list context for concept United include list item "*AA*", "*JetBlue*", "*Southwest*". Then we can calculate

Table 3.7: A list context example. $L1$, $L2$, $L3$ are three candidate lists. The list context for "*Delta*" is marked by underline *item*, the list context for "*United*" is marked by double-underline item.

L_1: AA, *Delta*, JetBlue
L_2: AA, *Delta*, JetBlue, Southwest
L_3: AA, *United*, JetBlue, Southwest

the similarity based on list context in the same way as *TextContextSim* (*i.e.*, using term frequency represent with cosine similar measure). This context similar feature is called *ListContextSim*. *ListContextSim* is to some extent similar to *ListCooccur*, but one advantage of *ListContextSim* is that even if two list items do not co-occur together in a candidate list, they may still have a high *ListContextSim*. For the example in Table 3.7, "*Delta*" and "*United*" do not co-occur together in a candidate list, but their list contexts are very similar, and thus they obtain a high *ListContextSim*.

3.6.3 Maximum Likelihood Parameter Estimation

We estimate the parameters λ, μ in the model by maximizing the conditional likelihood of a giving training set. (Later in Chapter 5, we will present another method for parameter estimation by directly maximizing the performing measure.) The training set for the graphical model can be denoted as $\{T_{\mathcal{L}}^{(k)}, P_{\mathcal{L}}^{(k)}, Y^{*(k)}, Z^{*(k)}\}$, where $Y^{*(k)}$, $Z^{*(k)}$ are the ground truth labels for the list items $T_{\mathcal{L}}^{(k)}$ and item pairs $P_{\mathcal{L}}^{(k)}$. (We use

superscript "*" to denote ground truth labels.) The conditional probability of the training set can be calculated according to Equation 3.4.

$$P(\lambda, \mu) = \prod_k P(Y^{*(k)}, Z^{*(k)} | T_{\mathcal{L}}^{(k)}, P_{\mathcal{L}}^{(k)}), \tag{3.4}$$

where $P(Y^{*(k)}, Z^{*(k)} | T_{\mathcal{L}}^{(k)}, P_{\mathcal{L}}^{(k)})$ is defined in Equation 3.3. Based on the condition probability, the conditional log-likelihood $l(\lambda, \mu)$, can be calculated as follows,

$$l(\lambda, \mu) = l_t(\lambda) + l_p(\mu), \tag{3.5}$$

$$l_t(\lambda) = \sum_k \sum_i \log P(y_i^{*(k)} | t_i^{(k)}) - \frac{\sum_k \lambda_k^2}{2\sigma^2}, \tag{3.6}$$

$$l_p(\mu) = \sum_k \sum_{i,j} \log P(z_{i,j}^{*(k)} | p_{i,j}^{(k)}, y_i^{*(k)}, y_j^{*(k)}) - \frac{\sum_k \mu_k^2}{2\gamma^2}, \tag{3.7}$$

where the last terms in Equation 3.6 and Equation 3.7 are served as regularizers, which penalize large values of λ, μ. σ and γ are regularization parameters that control the strength of penalty.

Notice that, in the train set, for those item pairs $p_{i,j}$ with any of its list item not being a facet term, their labels $z_{i,j}^* = 0$. According to Equation 3.2, for those item pairs, $\log P(z_{i,j}^* | p_{i,j}, y_i^*, y_j^*) = 0$, which makes no contribution to the conditional log-likelihood $l(\mu)$, and thus $l_p(\mu)$ can be simplified as

$$l_p(\mu) = \sum_k \sum_{i,j: y_i^{*(k)}=1, y_j^{*(k)}=1} \log P(z_{i,j}^{*(k)} | p_{i,j}^{(k)}, y_i^{*(k)}, y_j^{*(k)}) - \frac{\sum_k \mu_k^2}{2\gamma^2}, \tag{3.8}$$

where the i, j now indexes only item pairs with both of its list items being facet terms ($i.e., y_i^{*(k)} = 1, y_j^{*(k)} = 1$).

We can see that Equations 3.6 and 3.8 are exactly the same as log-likelihoods for two separated logistic regressions. In fact, Equation 3.6 learns a logistic regression

model for whether a list item is a facet term, and Equation 3.8 learns a logistic regression model for whether two facet terms should be grouped together. The parameter λ and μ can be learned by maximizing the log-likelihood using gradient descent, exactly same as in logistic regression.

3.6.4 Inference

When given a new labeling task, we could perform maximum a posteriori inference - compute the most likely labels Y, Z by maximizing the joint conditional probability $P(Y, Z|T_{\mathcal{L}}, P_{\mathcal{L}})$. After that, the query facet set $\mathcal{F}$ can be easily induced from the labeling Y, Z. (Collect list items with $y_i = 1$ as facet terms, and group any two of them into a query facet if the corresponding $z_{i,j} = 1$.) Note that the graphical model we designed does not enforce the labeling to produce strict partitioning for facet terms. For example, when $Z_{1,2} = 1$, $Z_{2,3} = 1$, we may have $Z_{1,3} = 0$. Therefore, the labeling results may induce an overlapping clustering. Unfortunately, this optimization problem is NP-hard, which can be proved by a reduction from the Ising model (Barahona, 1982).

To facilitate developing solutions, we add the strict partitioning constraint that each facet term belongs to exactly one query facet. Also, to directly produce the query facets, instead of inducing them after predicting labels, we rephrase the optimization problem as follows. First, we use the following notations for log-likelihoods,

$$\begin{aligned}
s_t(t_i) &= \log P(y_i = 1|t_i) \\
\overline{s_t}(t_i) &= \log\left(1 - P(y_i = 1|t_i)\right) \\
s_p(t_i, t_j) &= \log P(z_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1) \\
\overline{s_p}(t_i, t_j) &= \log\left(1 - P(p_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1)\right)
\end{aligned}$$

Using the notations above, the log-likelihood $l(\mathcal{F})$ for a particular query facet set $\mathcal{F}$ formed from $\mathcal{L}$ can be written as

$$
\begin{aligned}
l(\mathcal{F}) &= l_t(\mathcal{F}) + l_p(\mathcal{F}) \\
l_t(\mathcal{F}) &= \sum_{t \in T_{\mathcal{F}}} s_t(t_i) + \sum_{t \notin T_{\mathcal{F}}} \overline{s_t}(t_i) \\
l_p(\mathcal{F}) &= \sum_{F \in \mathcal{F}} \sum_{t_i, t_j \in F} s_p(t_i, t_j) + \sum_{\substack{F, F' \\ \in \mathcal{F}}} \sum_{\substack{t_i \in F, \\ t_j \in F'}} \overline{s_p}(t_i, t_j)
\end{aligned} \tag{3.9}
$$

In the right hand side of Equation 3.9, the first term is the intra-facet score, which sums up $s_p(\cdot, \cdot)$ for all the item pairs in each query facet. The second term is the inter-facet score, which sums up the $\overline{s_p}(\cdot, \cdot)$ for each item pair that appears in different query facets. Then the optimization target becomes $\mathcal{F} = \arg\max_{\mathcal{F} \in \mathfrak{F}} l(\mathcal{F})$, where $\mathfrak{F}$ is the set of all possible query facet sets that can be generated from $\mathcal{L}$ with the strict partitioning constraint.

This optimization problem, however, is still NP-hard, which can be proved by a reduction from the Multiway Cut problem (Bansal et al., 2002). Therefore, we propose two algorithms, **QFI** and **QFJ**, to approximate the results.

3.6.4.1 QFI

QFI (Query Faceting Independent) approximates the results by predicting whether a list item is a facet term and whether two list items should be grouped in a query facet independently, which is accomplished in two phases. In the first phase, QFI selects a set of list items as facet terms $T_{\mathcal{F}}$ according to $P(y_i|t_i)$. In this way, the algorithm predicts whether a list item t_i is a facet term independently, ignoring the dependences between y_i and its connected variables in Z. In our implementation, we simply select list items t_i with $P(t_i) > w_{min}$ as facet terms. (For convenience, we use $P(t_i)$ to denote $P(y_i = 1|t_i)$.) In the second phase, the algorithm clusters the facet terms $T_{\mathcal{F}}$ selected in the first phase into query facets, according to $P(t_i, t_j)$, where $P(t_i, t_j)$ is used to denote $P(z_{i,j} = 1|p_{i,j}, y_i = 1, y_j = 1)$. Using $P(t_i, t_j)$ as the distance measure, many clustering algorithm can be applied here. For our implementation, we use a cluster algorithm based on WQT (Quality Threshold with Weighted items) (Dou

et al., 2011), because it considers the importance of terms while clustering. We use $P(t_i)$ as the measure for facet term importance, and $dist_t(t_i, t_j) = 1 - P(t_i, t_j)$ as the distance measure for facet terms. The distance between a cluster and a facet term is computed using complete linkage distance, $dist_f(F, t) = max_{t' \in F} d_t(t, t')$, and the diameter of a cluster can be calculated as $dia(F) = \max_{t_i, t_j \in F} dist_t(t_i, t_j)$.

Algorithm QFI is shown in Algorithm 1. It takes the thresholds for term probability w_{min} and cluster diameter dia_{max} as inputs, as well as measure functions for term importance $P(t)$, distance $dist_f(F, t)$ and cluster diameter $dia(F)$. First, it selects facet terms by thresholding $P(t)$ (Line 1). Then it initializes the facet term pool to be clustered (Line 2), and sets the return result as an empty set (Line 3). After that, it processes the facet terms in decreasing order of $P(t)$, and builds a cluster for each of them (Line 4 to 17). For each facet term, it builds a cluster by iteratively adding the facet term that is closest to the cluster (Line 9), until the diameter of the cluster surpasses the threshold dia_{max} (Line 10). Last, clusters are collected (Line 16) and returned (Line 18).

QFI contains two parameters w_{min}, dia_{max}. They can be tuned for optimizing a given performance measure. QFI is also efficient. The dominant procedure for time complexity is the clustering part (Line 4 to 17). In clustering, each facet term is considered only once for forming a cluster or being added to a cluster. When considering adding a facet term to a cluster, it takes $O(m)$ time to calculate the distance $dist_f(F, t)$, where m is the number of facet terms in the F. m is bounded by the total number of facet terms selected, n. Therefore, overall QFI takes $O(n^2)$ time.

3.6.4.2 QFJ

QFI finds query facets based on the graphical model by performing inference of y_i and $z_{i,j}$ independently. The second algorithm, QFJ (Query Faceting Joint), instead tries to perform joint inference by approximately maximizing our target $l(\mathcal{F})$ with

Algorithm 1 QFI

Input: $w_{min}, dia_{max}, P(t), dist_f(F,t), dia(F)$
Output: $\mathcal{F}$

```
1: T_F ← {t | P(t) > w_min}
2: T_pool ← T_F
3: F ← ∅
4: while T_pool ≠ ∅ do
5:     t ← argmax_{t∈T_pool} P(t)
6:     F ← {t}
7:     T_pool ← T_pool − {t}
8:     while T_pool ≠ ∅ do
9:         t' ← argmin_{t'∈T_pool} dist_f(F, t')
10:        if dia(F ∪ {t'}) > dia_max then
11:            break
12:        end if
13:        F ← F ∪ {t'}
14:        T_pool ← T_pool − {t'}
15:    end while
16:    F ← F ∪ {F}
17: end while
18: return F
```

respect to y_i and $z_{i,j}$ iteratively. The algorithm first guesses a set of list items as facet terms. Then it clusters those facet terms by approximately maximizing $l_p(\mathcal{F})$, using a greedy approach. After clustering, the algorithm checks whether each facet term "fits" in its cluster, and removes those that do not fit. Using the remaining facet terms, the algorithm repeats the process (clustering and removing outliers) until convergence.

QFJ is outlined in Algorithm 2. The inputs to the algorithm include the candidate list item set $T_{\mathcal{L}}$, term probability $P(t)$, the log-likelihoods $l(\mathcal{F})$, $l_p(\mathcal{F})$. First, we select top n list items according to $P(t)$ as the initial facet terms (Line 1), because it is less sensitive to the absolute value of the probabilities. In our experiment, n is set to 1000 to make sure most of the correct facet terms are included. Then, the algorithm optimizes $l(\mathcal{F})$ by iteratively performing functions Cluster and RemoveOutliers.

Cluster performs clustering over a given set of facet terms. In Line 10 to 14, it puts each facet terms into a query facet by greedily choosing the best facet, or

Algorithm 2 QFJ

Input: $T_{\mathcal{L}}, P(t), l(\mathcal{F}), l_p(\mathcal{F}), n$
Output: $\mathcal{F} = \{F\}$

```
 1: T_F ← top n list items from T_L according to P(t)
 2: repeat
 3:     F ← CLUSTER(T_F, l_p)
 4:     T_F ← REMOVEOUTLIERS(F, l)
 5: until converge
 6: return F
 7:
 8: function CLUSTER(T_F, l_p)
 9:     F ← ∅
10:     for each t ∈ T_F in decreasing order of P(t) do
11:         choose to put t into the best facet in F
12:         or add t as a singleton into F,
13:         whichever has the highest resulting l_p(F).
14:     end for
15:     return F
16: end function
17:
18: function REMOVEOUTLIERS(F, l)
19:     T_F ← all facet terms in F
20:     for each F ∈ F do
21:         F' = ∅
22:         for each t ∈ F in decreasing order of P(t) do
23:             choose to add t into F' or not,
24:             whichever has the highest resulting l({F'})
25:             if not, T_F ← T_F − {t}
26:         end for
27:     end for
28:     return T_F
29: end function
```

creates a singleton for the list item, according to the resulting log-likelihood, $l_p(\mathcal{F})$. We choose to process these list items in decreasing order of $P(t)$, because it is more likely to form a good query facet in the beginning by doing so.

REMOVEOUTLIERS removes outlier facet terms in the clusters according to the joint log-likelihood $l(\mathcal{F})$. Initially, $T_{\mathcal{F}}$ contains all facet terms in $\mathcal{F}$ (Line 1). Then, for each cluster in $\mathcal{F}$, the function checks each facet term inside it (Line 22) to see if the facet term fits in the cluster according to the likelihood $l(\{F'\})$. If the facet term

does not fit in the cluster (*i.e.*, the facet term is an outlier), the function removes it from $T_{\mathcal{F}}$ (Line 25). Last, the function returns the remaining facet terms in $T_{\mathcal{F}}$ (Line 28), which will be re-clustered in CLUSTER.

The parameter n is set to a large number (1000 in our experiments) to ensure not missing facet terms initially, and thus it is not used as a tuning parameter. Therefore, different from QFI, QFJ does not have tuning parameters for optimizing a given performance measure. Instead, QFJ is purely guided by the log-likelihood object in the inferencing procedure. In terms of time complexity QFJ is close to QFI. CLUSTER processes each of the n facet terms. For each facet term, it considers k potential query facets to add the facet term. For each query facet option, the computation for the resulting $l_p(\mathcal{F})$ takes $O(n)$ time. Thus, overall CLUSTER takes $O(kn^2)$ time. REMOVEOUTLIERS considers to remove each of the n facet term as an outlier only once. In considering removing a facet term, computing $l\{F'\}$ takes $O(n)$ time. Thus, the overall time complexity for REMOVEOUTLIERS is $O(n^2)$. QFJ repeats CLUSTER and REMOVEOUTLIERS until convergence, thus QFJ takes $O(r \cdot kn^2)$ time, where r is the number of iterations. In practice, we find r and k are small. Thus, the time complexity for QFJ is similar to QFI.

3.6.5 Ranking Query Facets

The output of QFI and QFJ is a set of query facets $\mathcal{F}$. To produce ranking results, we defined a score for a query facet as $score_F(F) = \sum_{t \in F} P(t)$, and rank the query facets according to this scoring, in order to present more facet terms in the top. Facet terms within a query facet are ranked according to $score_t(t) = P(t)$. We leave the investigation for more sophisticated ranking models as future work.

3.7 Other Approaches

In this section, we describe two alternative method for facet refining. They are used as baselines in our experiments.

3.7.1 QDMiner

Dou et al. (2011) developed QDMiner/QDM for query facet extraction, which appears to be the first work that addressed the problem of query facet extraction. To solve the problem of finding query facets from the noisy candidate lists extracted, they used an unsupervised clustering approach. It first scores each candidate list by combining some TF/IDF-based scores. The candidate lists are then clustered with bias toward important candidate lists, using a variation of the Quality Threshold clustering algorithm (Heyer et al., 1999). After clustering, clusters are ranked and list items in each clusters are ranked/selected based on some heuristics. Finally, the top k clusters are returned as results. This unsupervised approach does not gain by having human labels available. Also, by clustering lists, they lose the flexibility of breaking a candidate list into different query facets.

3.7.2 Topic modeling

In semantic class extraction, Zhang et al. (2009) proposed to use topic models to find high-quality semantic classes from a large collection of extracted candidate lists. Their assumption is, like documents in the conventional setting, candidate lists are generated by a mixture of hidden topics, which are the query facets in our case. pLSA and LDA are used in their experiments. We find this approach can be directly used for finding query facets from candidate lists. The major change we need to make is that: in semantic class extraction, topic modeling is applied globally on the candidate lists (or a sample of them) from the entire corpus; in query facet extraction, we apply topic modeling only on the top k search results $\mathcal{D}$, assuming the coordinate terms in $\mathcal{D}$ are relevant to the query. Then, the topics are returned as query facets, by

using the top n list items in each topic (according to the list item's probability in the topic). Though this topic modeling approach is more theoretically motivated, it does not have the flexibility of adding different features to capture different aspects such as query relevance.

3.8 Summary

In this chapter, we developed query facet extraction, which extracts facets for a given query from its search results. We developed a supervised approach based on a graphical model to recognize facets from the noisy candidates found. The graphical model learns how likely a candidate term is to be a facet term as well as how likely two terms are to be grouped together in a query facet, and captures the dependencies between the two factors. We proposed two algorithms (QFI and QFJ) for approximate inference on the graphical model since exact inference is intractable. Compared with other existing methods, our models can easily incorporate a rich set of features, and learn from available labeled data.

To evaluate our models, in the next chapter, we develop an intrinsic evaluation method that compares extracted facets with human created ones.

CHAPTER 4

INTRINSIC EVALUATION

4.1 Introduction

Intrinsic evaluation is to evaluate the quality of query facet generation itself. We perform intrinsic evaluation by comparing system generated query facets with "gold standard" query facets. Note that this evaluation can be easily extended for evaluating new models (*i.e.*, compare the query facets generated by the new models with the existing "gold standard" query facets collected before). Later, in Chapter 7, we will carry out an extrinsic evaluation that evaluates the quality of generated facets by their utility in assisting search.

The "gold standard" query facets are constructed by human annotators and used as the ground truth to be compared with facets generated by different systems. The facet annotation is usually done by first pooling facets generated by different systems. Then annotators are asked to group or re-group terms in the pool into preferred query facets, and to give ratings for each of them regarding how useful or important the facet is.

In intrinsic evaluation, the quality of generated query facets can be measured from two aspects. (1) How well does the model generate/find correct facet terms. This can be measured by standard classification measures, such as precision and recall. (2) How well does the model groups facet terms correctly. This can be measured by standard clustering measures, such as F-measures for clustering, Purity and Normalized Mutual Information.

However, a single performance measure that combines the different aspects can be desirable in many cases. First, such a single measure provides an overall measurement for effectiveness, which is often necessary for comparing different models. Second, such a single measure can be used for tuning or training models. For example, in Chapter 5, we propose a method that directly uses a performance measure as the training objective.

To combine the two evaluation aspects, we design a new measure called $PRF_{\alpha,\beta}$ (Kong and Allan, 2013). $PRF_{\alpha,\beta}$ combines TP, TR and PF using weighted harmonic mean, where TP, TR and PF are precision and recall for facet terms, and the F1 measure for facet term clustering. Parameters α and β can be used to adjust the emphasis of the three factors for different applications.

However, $PRF_{\alpha,\beta}$ does not directly account for facet ranking performance. Dou et al. (2011) used some variations of nDCG to evaluate facet ranking. In the nDCG variation measures, system facets are mapped to truth facets, and assigned ratings according to their mapped truth facets. Then the ranked system facets are evaluated using nDCG, with the discounted gain further weighted by the precision and recall of the system facet and mapped truth facet. However, we will show that this metric can be problematic in some cases.

In the experiments of this chapter, we perform intrinsic evaluation on the different query facet extraction models described in Chapter 3. The experimental results show that our supervised methods (QFI and QFJ) described in Chapter 3 significantly outperform other unsupervised methods, suggesting that query facet extraction can be effectively learned.

In the rest of this chapter, we first describe how we collect data and perform facet annotation for intrinsic evaluation in Section 4.2. Then, we describe different evaluation metrics in Section 4.3, including $PRF_{\alpha,\beta}$ and other existing measures. In Section 4.4, we present our experiments for comparing different query facet extraction

models, analyzing the features used in QFI/QFJ, and testing different candidate extraction patterns. Last, we summarize this chapter in Section 4.5.

4.2 Data

4.2.1 Query

We constructed a pool of 232 queries from different sources, including random samples from a query log, TREC 2009 Web Track queries[1], example queries appearing in related publications (Xue and Yin, 2011; Wang et al., 2009) and queries generated by our annotators. Annotators were asked to select queries that they are familiar with from the pool for annotating. Overall, we collect annotations for 100 queries (see Table 4.1).

Table 4.1: Query statistics

Source	#queries collected	#queries annotated
Query log	100	30
Related publications	20	10
TREC 2009 Web Track	50	20
Annotators generated	62	40
Sum	232	100

4.2.2 Search Results

For each query, we acquire the top 100 search results from the commercial search engine Bing[2] during December, 2012. A few search results were skipped due to crawl errors, or if they were not HTML Web pages. For the 232-query set, we crawled 22,909 web pages, used for extracting feature $listIDF$ described in Section 3.6.2.4.

[1] http://trec.nist.gov/data/web/09/wt09.topics.queries-only

[2] http://www.bing.com/

For the 100 annotated queries, the average number of crawled web pages is 98.7, the minimum is 79, both the maximum and the median are 100.

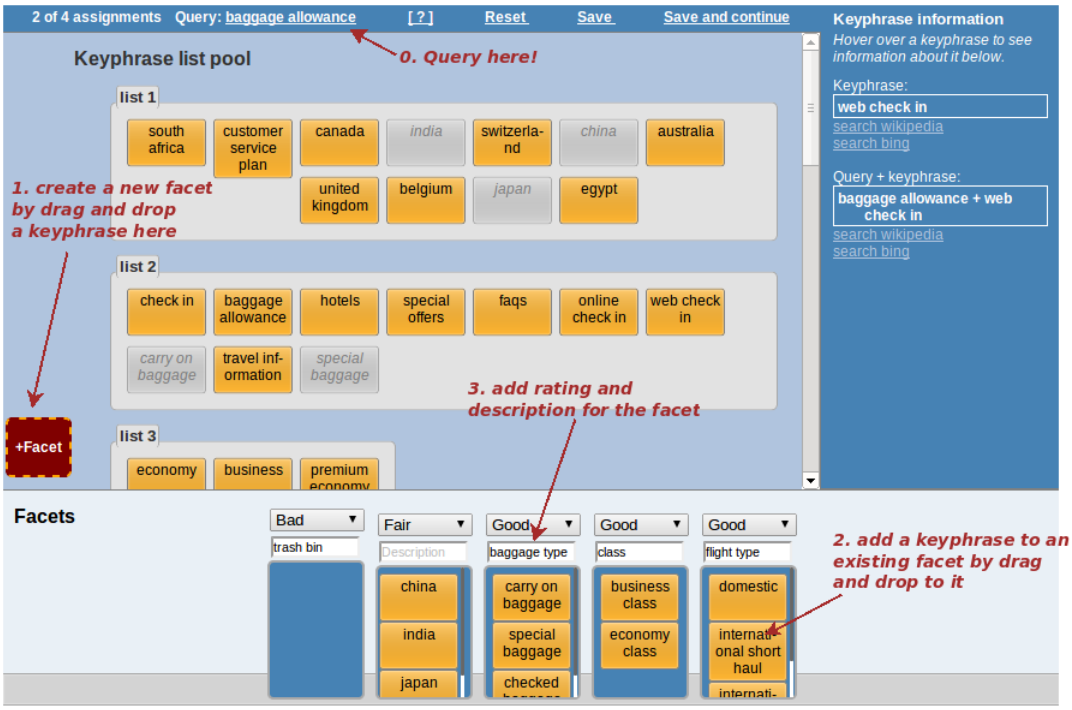


Figure 4.1: Annotation interface for query facet annotation.

4.2.3 Query facet annotations

We asked human annotators to construct query facets as ground truth, using the annotation interface shown in Figure 4.1. For each query, we first constructed a pool of terms by aggregating facet terms in the top 10 query facets generated by different models (corresponding to "keyphrase list pool" in Figure 4.1), including two runs from QDM, one run from each of pLSA and LDA using top 10 list items in each query facets, and one run for our graphical model based approach. Then, annotators were asked to group terms in the pool into query facets for each query they selected (corresponding to step 1 and 2 in Figure 4.1). Finally, the annotator was asked to give a rating for each constructed query facet, regarding how useful and important the query facet is (corresponding to step 3 in Figure 4.1). The rating scale of good=2/fair=1 is used.

The facet with rating "bad" in Figure 4.1 serves as a trash bin (for deleting terms). All facets rated as bad are not used in our study.

We recruited 17 human annotators (5 females and 12 males), with 16 graduate students (computer science major) and 1 undergraduate student from our university. Prior to the actual annotation task, annotators were assigned a training session (using another query that is not in the data set) to get familiar with the task and annotation interface.

There are 50 query facets pooled per query, with 224.8 distinct facet terms per query. Annotation statistics for the good and fair facets, as well as the pooled facet, are given in Table 4.2. The table shows average number of facet terms per query, average number of query facets per query, and average number of facet terms per facet, for each categories (fair, good, and pooled facets).

Table 4.2: Annotation statistics

	fair	good	pooled
#terms per query	26.6	55.8	224.8
#facets per query	3.1	4.8	50.0
#terms per facet	8.6	11.6	8.8

4.3 Metrics

Query facet extraction can be evaluated from different aspects. A good system should select "correct" facet terms from all the list items, therefore we use standard classification metrics, such as precision, recall and F-measures. A good system should also group those facet term correctly (i.e., in the same way as the annotators), therefore we use standard clustering metrics, such as F-measures for clustering, purity and normalized mutual information. To combine the different evaluation aspects, we design a new measure for this particular task.

4.3.1 Notations

We continue to use notation defined in Section 3.6.1 for a query facet (F), a query facet set ($\mathcal{F}$) and the set of all facet terms ($T_{\mathcal{F}}$) in a query facet set. We use "$*$" to distinguish between system generated results and human labeled results, which we used as ground truth. For example, $\mathcal{F}$ denotes the system generated query facet set, and $\mathcal{F}^*$ denotes the human labeled query facet set. For convenience, we use T to denote $T_{\mathcal{F}}$ in this Chapter, omitting the subscript $\mathcal{F}$. T^* denotes all the facet terms in the human labeled query facet set. We use r_{F^*} to denote the rating score for a human labeled facet F^*.

4.3.2 Effectiveness in finding facet terms

One aspect of query facet extraction evaluation is how well a system finds facet terms. This can be evaluated using standard classification metrics as follows,

- facet term precision: $TP = \frac{|T \cap T^*|}{|T|}$
- facet term recall: $TR = \frac{|T \cap T^*|}{|T^*|}$
- facet term F1: $TF = \frac{2|T \cap T^*|}{|T|+|T^*|}$

where the "T" in measure names TP, TR, TF stands for facet *term*. It is used to distinguish the term based measures from term pair based measures as defined below. Note that these metrics do not take clustering quality into account.

4.3.3 Clustering quality

To evaluate how well a system groups facet terms correctly, similar to Dou et al. (2011), we use several existing cluster metrics, namely, purity, NMI/normalized mutual information and pair-counting F1 measure. Here the pair-counting F1 measure treats term clustering as classification on whether each pair of terms is in the same facet, and then combines pair precision and recall using the F1 measure. We denote the pair-counting F1 measure as PF with "P" standing for term *pair*.

In our task, we usually have $T \neq T^*$. The facet terms in the system-generated and human-labeled clustering results might be different: the system might fail to include some human identified facet terms, or it might mistakenly include some "incorrect" facet terms. These standard clustering metrics cannot handle these cases properly. To solve this problem, we adjust $\mathcal{F}$ and $\mathcal{F}^*$ as if only facet terms in $T \cap T^*$ were clustered by the system, since we are only interested in how well the "correct" facet terms are clustered from these metrics. The adjusting is done by removing "incorrect" facet terms ($t \in T - T^*$) from $\mathcal{F}$, and removing missing facet terms ($t^* \in T^* - T$) in $\mathcal{F}^*$. By this adjusting, we do not take into account the effectiveness of finding correct facet terms. The effectiveness of finding correct facet terms has already been captured by our classification measures, TP, TR and TF.

4.3.4 Overall quality

To evaluate the overall quality of query facet extraction, Dou et al. (2011) proposed variations of nDCG (Normalized Discounted Cumulative Gain), namely purity-aware nDCG (pNDCG, abbreviated as fp-nDCG in the original paper) and recall- and purity-aware nDCG (prNDCG, abbreviated as rp-nDCG in the original paper). They first map each system generated facet F to a human labeled facet F^* that covers the maximum number of terms in F. Then, they assign the rating r_{F^*} to F, and evaluate $\mathcal{F}$ as a ranked list of query facets using nDCG. The discounted gains are weighted by precision and/or recall of facet terms in F, against its mapped human labeled facet F^*. For **pNDCG**, only precision is used as the weight, $\frac{|F^* \cap F|}{|F|}$. For **prNDCG**, precision and recall are multiplied as the weight, $\frac{|F^* \cap F|^2}{|F^*||F|}$. One other way is to use F1 for weighting as $\frac{2|F^* \cap F|}{|F^*|+|F|}$, and we call this measure **fNDCG**.

However, these nDCG variation measures can be problematic in some cases. When two facets F_1 and F_2 are mapped to the same human labeled facet F^*, only the first facet F_1 is credited and F_2 is simply ignored, even if it is more appropriate to map

F_2 to F^* (e.g., F_2 is exactly same as F^*, while F_1 contain only one facet term in F^*). Our proposed metric does not need to map facets, and thus does not have this problem.

The quality of query facet extraction is intrinsically multi-faceted. Different applications might have different emphasis in the three factors mentioned above - precision of facet terms, recall of facet terms and clustering quality of facet terms. We propose a metric $PRF_{\alpha,\beta}$ to combine the three factors, using weighted harmonic mean, as follows

$$PRF_{\alpha,\beta}(TP, TR, PF) = \frac{(\alpha^2 + \beta^2 + 1)}{\frac{\alpha^2}{TP} + \frac{\beta^2}{TR} + \frac{1}{PF}}, \tag{4.1}$$

where $\alpha, \beta \in [0, +\infty)$ are used to control the weight between the three factors in the same way as "β" in F-measures (van Rijsbergen, 1979). α and β can be interpreted as the importance of TP and TR compared to PF respectively. More formally, we have

$$\begin{aligned} when\ \alpha &= \frac{TP}{PF}, \frac{\partial PRF_{\alpha,\beta}}{\partial TP} = \frac{\partial PRF_{\alpha,\beta}}{\partial PF} \\ when\ \beta &= \frac{TR}{PF}, \frac{\partial PRF_{\alpha,\beta}}{\partial TR} = \frac{\partial PRF_{\alpha,\beta}}{\partial PF}. \end{aligned} \tag{4.2}$$

The intuition behind this is we want to specify the TP/PF ratio at which the user is willing to trade an increment in TP for an equal loss in PF, and similarly for TR/PF. For example, we can set $\alpha = 2, \beta = 1$ to evaluate the case where TP is twice as important as TR and PF. When $\alpha{=}\beta{=}1$, we omit the subscript part for simplicity, i.e. $PRF \equiv PRF_{1,1}$.

While $PRF_{\alpha,\beta}$ has the flexibility to adjust emphasis between the three factors, it does not take into account the different ratings associated with query facets. To incorporate ratings, we use a weighted version of TP, TR and PF in $PRF_{\alpha,\beta}$. We call the new metric $wPRF_{\alpha,\beta}$. The weighted facet term precision, recall and TF are defined as follows.

- weighted facet term precision: $wTP = \frac{\sum_{t \in T \cap T^*} w(t)}{\sum_{t \in T} w(t)}$
- weighted facet term recall: $wTR = \frac{\sum_{t \in T \cap T^*} w(t)}{\sum_{t^* \in T^*} w(t^*)}$
- weighted facet term F1: $wTF = \frac{2wP(T,T^*)wR(T,T^*)}{wP(T,T^*)+wR(T,T^*)}$

where $w(t)$ is the weight for facet term t, and assigned as follows

$$w(t) = \begin{cases} r_{F^*} & \text{if } t \in T^* \\ 1 & \text{otherwise} \end{cases}$$

Similarly, wPF is computed by weighting its pairwise precision and recall in the same fashion as the weighted facet term precision and recall above. Instead of $w(t)$, we need weight for a pair of facet terms $w(t_1, t_2)$ in this calculation. We assign weight for facet term pair $w(t_1, t_2)$ using their sum, $w(t_1) + w(t_2)$.

4.4 Experiments

In this section, we conduct experiments based on intrinsic evaluation described above. In the experiments, we investigate different facet refining models, analyze features used in our query faceting models (Section 3.6.2.4) and different facet candidate extraction patterns (Section 3.4).

4.4.1 Experiment settings

We compare the effectiveness of the five models, QDM, pLSA, LDA and QFI, QFJ (described in Chapter 3), on the 100-query data set, which we call QF13. All the models take the same extracted and cleaned candidate lists (see Section 3.4) as input. We perform 10-fold cross validation for training/testing and parameter tuning (if applicable) in all experiments and for all models. When training the graphical model, we standardize features (*i.e.*, determine the distribution mean and standard deviation for each feature from the train data set. Then subtract the mean from each

feature. Then divide the values of each feature by its standard deviation.) We set both of the two regularizers σ and γ to be 1. There are too many negative instances ($y_i = 0,\ z_{i,j} = 0$) in the training data, so we stratify samples by labels with the ratio of positive:negative to be 1:3. For QDM, we tune the two parameters used in the clustering algorithm Dia_{max} (the diameter threshold for a cluster) and W_{min} (the weight threshold for a valid cluster), as well as two parameters used for selecting facet terms in each facet ($S_{t|F} > \alpha|Sites(F)|$ and $S_{t|F} > \beta$). For pLSA and LDA, we tune the number of facet terms in a query facet. For QFI, we tune the weight threshold for facet terms, w_{min}, and the diameter threshold, dia_{max}. For QFJ, there are no parameter that need to be tuned. We returned top 10 query facets from all the five models in all evaluation.

4.4.2 Finding Facet Terms

We first evaluate the models in terms of their effectiveness in finding facet terms. More specifically, we compare the classification performance of these models in terms of term precision (TF), term recall (TR), term F1 (TF) and their weighted versions (wTF, wTR, wTF). We compare results tuned on TF, which keeps a balance between term precision and term recall, and wTF, which, in addition, takes facet term weighting into account.

We report the results in Table 4.3. From the table, we can see that QFI and QFJ perform relatively well for both precision and recall. The two topic model based approaches, pLSA and LDA, have relatively high recall and low precision. Contrarily, QDM has higher precision than the two topic models, but low recall. This difference can be explain by the number of facet terms each model returned, as shown in the last column of the table. QDM only outputs 93.4 facet terms per query, while pLSA and LDA both output many more facet terms. One possible reason for the low precision of pLSA and LDA is that they select facet terms solely according to term probabilities

Table 4.3: Facet term classification performance. Results in the upper part are tuned on TF, and results in the bottom part are tuned on wTF. "#terms" shows the average number of facet terms returned per query for each models. The best performance scores are marked in boldface. Statistically significant improvements ($p < 0.05$, paired t-test) of QFI/QFJ over other models are marked by superscripts/subscripts q for QDM, p for pLSA, l for LDA and j for QFJ.

Tuned on TF							
Model	TP	TR	TF	wTP	wTR	wTF	#terms
QDM	0.3124	0.3182	0.2926	0.2773	0.2852	0.2604	93.4
pLSA	0.2627	**0.5640**	0.3350	0.2305	0.5001	0.2950	175.0
LDA	0.2743	0.5382	0.3365	0.2385	0.4743	0.2941	154.0
QFJ	$0.3986^{q}_{p,l}$	0.4832^{p}	$0.4161^{q}_{p,l}$	$0.3482^{q}_{p,l}$	0.4267^{p}	$0.3650^{q}_{p,l}$	97.0
QFI	$\mathbf{0.4157}^{q}_{p,l}$	$0.5543^{q,j}$	$\mathbf{0.4472}^{q,j}_{p,l}$	$\mathbf{0.3712}^{q,j}_{p,l}$	$\mathbf{0.5018}^{q,j}$	$\mathbf{0.4017}^{q,j}_{p,l}$	107.9
Tuned on wTF							
Model	TP	TR	TF	wTP	wTR	wTF	#terms
QDM	0.3124	0.3182	0.2926	0.2773	0.2852	0.2604	93.4
pLSA	0.2627	0.5640	0.3350	0.2305	0.5001	0.2950	175.0
LDA	0.2625	**0.5936**	0.3389	0.2301	**0.5285**	0.2988	180.0
QFJ	$0.3986^{q}_{p,l}$	0.4832^{p}	$0.4161^{q}_{p,l}$	$0.3482^{q}_{p,l}$	0.4267^{p}	$0.3650^{q}_{p,l}$	97.0
QFI	$\mathbf{0.4058}^{q}_{p,l}$	$0.5670^{q,j}$	$\mathbf{0.4461}^{q,j}_{p,l}$	$\mathbf{0.3623}^{q}_{p,l}$	$0.5126^{q,j}$	$\mathbf{0.4003}^{q,j}_{p,l}$	112.6

in the learned topics (query facets in our case) and do not explicitly incorporate query relevance. We find most of their facet terms are frequently-occurring list items, which are not necessary relevant to the query. While the numbers of facet terms QFI and QFJ output are similar to QDM, QFI and QFJ obtain much higher precision and recall, likely due to the rich set of features used, which captures both how likely a list item is to be a coordinate term, and how likely it is to be relevant to the query. We will analyze these features in Sections 4.4.5.

From Table 4.3, we also find the the weighted measures are usually consistent with their corresponding unweighted measures.

4.4.3 Clustering Facet Terms

Next, we evaluate the models in terms of their effectiveness in clustering facet terms, using clustering measures described in Section 4.3.3. More specifically, we

compare the clustering performance of these models in terms of pair counting F1 (PF), wPF (weighted version of PF), purity and NMI (normalized mutual information). We compare results tuned on PF and wPF, which takes facet term weighting into account.

We report the results in Table 4.4. From the table, we can see that QFI and QFJ perform relatively well for PF, wPF and NMI. The improvements of QFI and QFJ over the other three models shown are all significant ($p < 0.05$, using paired t-test). For purity, pLSA and LDA obtain relatively higher score, which is because they returned a relatively small number of facet terms (shown in the "#terms" column), and thus only cluster a very small number of terms together in each clusters. These observations are consistent for the runs tuned on PF and its weighted version, wPF.

Table 4.4: Facet term clustering performance. Results in the upper part are tuned on PF, and results in the bottom part are tuned on wPF. We also report corresponding classification performance in the left part using term precision (TP), term recall (TR) and term F1 (TF). "#terms" shows the average number of facet terms returned per query for each models. The best performance scores are marked in boldface. Statistically significant improvements ($p < 0.05$, paired t-test) of QFI/QFJ over other models are marked by superscripts/subscripts q for QDM, p for pLSA, l for LDA and j for QFJ.

Tuned on PF								
Model	PF	wPF	Purity	NMI	TP	TR	TF	#terms
QDM	0.5543	0.5435	0.9484	0.5734	0.2489	0.2628	0.2206	103.9
pLSA	0.4270	0.4119	**0.9746**	0.5643	0.3355	0.1255	0.1646	28.9
LDA	0.3860	0.3657	0.9672	0.5616	0.3492	0.1388	0.1797	30.0
QFJ	$0.6961^{q}_{p,l}$	$0.6633^{q}_{p,l}$	0.9346	$0.6285^{q}_{p,l}$	$\mathbf{0.3986}^{q,i}_{p,l}$	$\mathbf{0.4832}^{q,i}_{p,l}$	$\mathbf{0.4161}^{q,i}_{p,l}$	97.0
QFI	$\mathbf{0.7397}^{q,j}_{p,l}$	$\mathbf{0.7130}^{q,j}_{p,l}$	$0.9628^{q,j}$	$\mathbf{0.6336}^{q}_{p,l}$	0.2063	$0.4052^{q}_{p,l}$	$0.2592^{q}_{p,l}$	161.5
Tuned on wPF								
Model	PF	wPF	Purity	NMI	TP	TR	TF	#terms
QDM	0.5831	0.5751	0.9772	0.5776	0.2236	0.1763	0.1636	92.7
pLSA	0.4327	0.4223	**0.9845**	0.5673	0.3398	0.0993	0.1437	21.8
LDA	0.4176	0.4007	0.9775	0.5646	0.3595	0.1132	0.1575	23.4
QFJ	$0.6961^{q}_{p,l}$	$0.6633^{q}_{p,l}$	0.9346	$0.6285^{q}_{p,l}$	$\mathbf{0.3986}^{q,i}_{p,l}$	$\mathbf{0.4832}^{q,i}_{p,l}$	$\mathbf{0.4161}^{q,i}_{p,l}$	97.0
QFI	$\mathbf{0.7370}^{q,j}_{p,l}$	$\mathbf{0.7090}^{q,j}_{p,l}$	0.9635^{j}	$\mathbf{0.6325}^{q,j}_{p,l}$	0.2055	$0.3993^{q}_{p,l}$	$0.2570^{q}_{p,l}$	159.5

The better performance in clustering for QFI and QFJ can be explained by their incorporating factors other than list item co-occurrence information. In our feature

analysis (Section 4.4.5), besides the one item co-occurrence related feature, *listContextSim*, we also find that *textContextSim* has a relatively high weight. *textContextSim* is used to capture the similarity of the two list items using their surrounding text, so it can help to group two facet terms together even if they might not co-occur a lot in candidate lists. As an example, for the query *baggage allowance*, we find different airlines do not co-occur a lot in candidate lists, (e.g. *delta* and *jetblue* only co-occur twice), but they tend to have high *textContextSim* (e.g. $TextContextSim(delta, jetblue) = 0.81$), and are therefore grouped together by QFI and QFJ.

From Table 4.4, we also find term clustering performance does not necessarily "agree" with term classification performance. Comparing QFI and QFJ, we find QFI obtains better clustering performance of the facet terms it selected, but does relatively poorly in selecting facet terms. Thus, next we will investigate the overall performance.

4.4.4 Overall Evaluation

To compare overall effectiveness of the five models, in this section, we focus on using $PRF_{\alpha,\beta}$ measure with equal weight between term precision, term recall and term clustering (denoted as PRF). We will investigate unbalanced weighting in $PRF_{\alpha,\beta}$ in Chapter 5.

We tune all the models on PRF, as well as its weighted version (wPRF). We report the overall measure scores (PRF, wPRF), as well as its constituent factors (TP, TR, PF and wTP, wTR, wPF), in order to see the details. We also report the rank measures, pNDCG, prNDCG and fNDCG, to see whether PRF based measures agree with these ranking measures. The results are given in Table 4.5.

Results in Table 4.5 are mostly consistent with the results that were tuned on TF and PF (and their weighted version) in the classification and clustering evaluation above. QDM obtain relatively low term precision and low term recall, but better clustering performance on the selected facet terms (PF and wPF). pLSA and LDA

Table 4.5: Overall performance tuned on PRF (upper) and wPRF (bottom). We also include term precision (TP), term recall (TR), term clustering pair counting F1 (PF), and their weighted versions (wTP, wTR and wPF). We also report ranking-based measures pNDCG, prNDCG and fNDCG, which weight the DCG gains by purity (or precision), recall and F1 of facet terms respectively. The best performance scores are marked in boldface. Statistically significant improvements ($p < 0.05$, paired t-test) of QFI/QFJ over other models are marked by superscripts/subscripts q for QDM, p for pLSA, l for LDA and j for QFJ.

Tuned on PRF								
Model	TP	TR	PF	PRF	pNDCG	prNDCG	fNDCG	#terms
QDM	0.2946	0.3284	0.5662	0.3279	0.1554	0.0564	0.1441	102.5
pLSA	0.2744	**0.5027**	0.4372	0.3411	0.1294	0.0508	0.1439	148.0
LDA	0.2802	0.4975	0.4018	0.3293	0.1307	0.0496	0.1411	138.6
QFJ	$0.3986^{q}_{p,l}$	0.4832^{q}	$\mathbf{0.6961}^{q,i}_{p,l}$	$0.4654^{q}_{p,l}$	$\mathbf{0.3256}^{q}_{p,l}$	$\mathbf{0.1771}^{q}_{p,l}$	$\mathbf{0.2946}^{q}_{p,l}$	97.0
QFI	$\mathbf{0.4450}^{q,j}_{p,l}$	0.4881^{q}	$0.6209^{q}_{p,l}$	$\mathbf{0.4720}^{q}_{p,l}$	$0.3176^{q}_{p,l}$	$0.1626^{q}_{p,l}$	$0.2857^{q}_{p,l}$	89.5
Tuned on wPRF								
Model	wTP	wTR	wPF	wPRF	pNDCG	prNDCG	fNDCG	#terms
QDM	0.2572	0.2967	0.5435	0.2941	0.1538	0.0565	0.1441	104.9
pLSA	0.2305	0.5001	0.3998	0.3062	0.1082	0.0500	0.1363	175.0
LDA	0.2287	**0.5276**	0.3686	0.2955	0.1065	0.0485	0.1298	180.0
QFJ	$0.3482^{q}_{p,l}$	0.4267^{q}	$\mathbf{0.6633}^{q,i}_{p,l}$	$0.4144^{q}_{p,l}$	$\mathbf{0.3256}^{q}_{p,l}$	$\mathbf{0.1771}^{q}_{p,l}$	$\mathbf{0.2946}^{q}_{p,l}$	97.0
QFI	$\mathbf{0.3897}^{q,j}_{p,l}$	0.4420^{q}	$0.5891_{p,l}$	$\mathbf{0.4263}^{q}_{p,l}$	$0.3167^{q}_{p,l}$	$0.1627^{q}_{p,l}$	$0.2858^{q}_{p,l}$	90.6

have high recall, but low precision and PF/wPF. This is due to that pLSA and LDA return a lot of facet terms. There are on average 81.15 facet terms per query for the human annotated query facets, but pLSA and LDA returned around twice of that number. QFI and QFJ are the best two models according to the overall performance measures, PRF, wPRF and pNDCG, prNDCG, fNDCG. The differences are statistically significant ($p < 0.05$ based on paired t-test). We will analysis more on QFI's and QFJ's success in Section 4.4.5.

Comparing the results tuned on PRF and its weighted version, wPRF, we find weighting encourages returning slightly more facet terms, which is shown in the column "*#terms*", the average number of facet terms returned per query. This can be explained by noting that returning more terms may increase the number of high-rated facet terms found, and thus increase wPRF. Comparing the results for PRF, wPRF

with the results for pNDCG, prNDCG, fNDCG, we find the two types of measures agree with each other in general (*e.g.*, QFJ and QFI are better than other three models), but not always (QFI is better than QFJ for the PRF measures, but worse than QFJ for the NDCG measures).

Since PRF/wPRF do not always agree with the ranking-based measures, and PRF/wPRF do not account for facet ranking effectiveness, we also test the models based on results tuned on these ranking-based measures. We report results for the ranking-based measures and PRF/wPRF tuned on fNDCG in Table 4.6 (results tuned on pNDCG, prNDCG are similar). QFI and QFJ are the best two models for the ranking-based measures. QFJ gives the best performance for prNDCG and fNDCG, which combine both precision and recall of facet terms in weighting DCG. These observations are consistent with the results tuned on PRF/wPRF in Table 4.5.

Table 4.6: Overall performance tuned on fNDCG. The best performance scores are marked in boldface. Statistically significant improvements ($p < 0.05$, paired t-test) of QFI/QFJ over other models are marked by superscripts/subscripts q for QDM, p for pLSA, l for LDA and j for QFJ.

Model	pNDCG	prNDCG	fNDCG	PRF	wPRF	#terms
QDM	0.1442	0.0550	0.1410	0.3331	0.2991	113.4
pLSA	0.1838	0.0674	0.1756	0.3303	0.2880	139.0
LDA	0.1718	0.0594	0.1588	0.3259	0.2844	145.5
QFJ	$0.3256^{q}_{p,l}$	$\mathbf{0.1771}^{q}_{p,l}$	$\mathbf{0.2946}^{q}_{p,l}$	$0.4654^{q}_{p,l}$	$0.4144^{q}_{p,l}$	97.0
QFI	$\mathbf{0.3350}^{q}_{p,l}$	$0.1618^{q,j}_{p,l}$	$0.2825^{q}_{p,l}$	$\mathbf{0.4678}^{q}_{p,l}$	$\mathbf{0.4197}^{q}_{p,l}$	77.5

4.4.5 Feature Analysis

In our analysis above, we credit the success of QFI/QFJ models to the rich set of features they used. In this section, we analyze these features to: (1) discover which features are important and which are not; (2) test our hypothesis that the success of QFI/QFJ is due to the rich set of features.

4.4.5.1 Feature Weight Analysis

We first test the importance of our features by their weight learned in the model. Note that our features have been standardized (Section 4.4.1), and thus the weights are comparable. Higher weight in its absolute suggests the corresponding feature is more important. Table 4.7 shows the weight learned for item features (Section 3.6.2.4) in one fold of the 10-fold cross-validation (results are similar for other folds). Not surprisingly, list TF/IDF based features which are used to capture the likelihood of being a coordinate term have relatively high weights, with *ListTermFreq.ListIDF* being the most important features. Other features that are used to capture query relevance also obtain relatively high weight, *e.g.*, *ContentSiteFreq*, *ContentTermFreq.IDF*.

In Table 4.8, we show the weights learned for item pair feature (Section 3.6.2.4) in one fold of the 10-fold cross-validation (results are similar for other folds). The table suggests that *ContextListSim* and *ContextTextSim* are the two most important item pair features. Though both *ContextListSim* and *ListCooccur* are based on the item occurrence in candidate lists, the weight for *ListCooccur* is far less than the weight for *ContextListSim*. This can be explained by the example in Table 3.7, which shows that *ContextListSim* can assign high value for semantically related list items, even if they do not co-occur in a candidate list.

4.4.5.2 Feature Ablation Experiments

Next we investigate the effectiveness of our features based on feature ablation experiments, in which we remove one feature (or a set of features) at a time to examine the effectiveness of the each feature (or feature set) in the presence of other features. The results are reported in Table 4.9.

From Table 4.9, we can see that the feature ablation experiments are in general consistent with our previous analysis based on feature weights. *ContextListSim*, *ListText* based features, *ListTermFreq.ListIDF*, *ContextTextSim* are the most informative

Table 4.7: Item feature weights learned in one fold of the 10-fold cross-validation (results are similar for other folds). Features are sorted by the absolute value of the weights. The features are explained in Table 3.5.

Feature	Weight
ListTermFreq.ListIDF	2.1620
ListSelectSiteFreq	1.9604
ContentSiteFreq	1.5251
ListTextSiteFreq	1.3860
ListSelectPageFreq	-1.0627
ListTrTermFreq	-0.8608
ListTdPageFreq	-0.8248
ContentTermFreq.IDF	0.8195
ContentWPageFreq	-0.7475
ListTrSiteFreq	0.7438
ListIDF	-0.6863
ListTdSiteFreq	0.6491
IDF	-0.6207
ListTextPageFreq	-0.5431
ContentPageFreq	0.4560
ContentTermFreq	-0.4193
ListUlPageFreq	0.2985
ListSelectTermFreq	-0.2908
ListTdTermFreq	0.2740
Length	-0.2615
TitleTermFreq	0.2497
ListUlSiteFreq	0.2438
ListUlTermFreq	-0.2345
ListOlSiteFreq	-0.1507
TitleSiteFreq	-0.1271
ListOlTermFreq	0.1203
ListOlPageFreq	0.1076
TitlePageFreq	-0.0447
ListTrPageFreq	-0.0424
ListTextTermFreq	-0.0295

features. Comparing the feature sets ListText, ListUl ListSelect, ListTd, ListTr, ListOl, we find features based on candidate lists extracted from TEXT, UL and SELECT patterns are more informative. In the next section, we will further investigate these different extraction patterns.

Table 4.8: Item pair feature weights learned in one fold of the 10-fold cross-validation (results are similar for other folds). Features are sorted by the absolute value of the weights. The features are explained in Table 3.6.

Feature	Weight
ContextListSim	1.5373
ContextTextSim	0.7754
ListCooccur	0.0643
LengthDiff	0.0271

Table 4.9: PRF performance changes when suppressing each features or feature sets. ΔPRF shows the PRF performance change when excluding the corresponding feature (or feature set). ΔPRF% shows the PRF change in percentage. The p-values reported are based on paired t-test. ListText, ListUl, ListSelect, ListOl, ListTr, ListTd, Content, Title denote feature sets in which the features are extracted from the corresponding fields (*e.g.*, ListText = {ListTextTermFreq, ListTextPageFreq, ListTextSiteFreq}). The results are based on QFJ model tuned on PRF (other results are similar).

Feature/Feature set	PRF	ΔPRF	ΔPRF%	p-value
ContextListSim	0.4416	-0.0238	-5.11%	6.0×10^{-4}
ListText	0.4454	-0.0200	-4.30%	3.0×10^{-5}
ListTermFreq.ListIDF	0.4465	-0.0189	-4.06%	3.2×10^{-4}
ContextTextSim	0.4531	-0.0123	-2.64%	0.0175
ListUl	0.4599	-0.0055	-1.18%	0.0873
ListSelect	0.4615	-0.0039	-0.84%	0.3798
LengthDiff	0.4616	-0.0038	-0.82%	0.2259
IDF	0.4620	-0.0034	-0.73%	0.2153
Content	0.4621	-0.0033	-0.71%	0.3655
ListTd	0.4633	-0.0021	-0.45%	0.4617
ListTr	0.4635	-0.0019	-0.41%	0.5814
Title	0.4638	-0.0016	-0.34%	0.5854
Length	0.4643	-0.0011	-0.24%	0.6626
ListIDF	0.4650	-0.0004	-0.09%	0.9048
ListCooccur	0.4653	-0.0001	-0.02%	0.9577
ContentTermFreq.ClueIDF	0.4657	0.0003	0.06%	0.9365
ListOl	0.4660	0.0006	0.13%	0.8130

4.4.5.3 Feature Accumulation Experiments

Last, we want to test our hypothesis that the improvements of QFI and QFJ over other models are due to the rich set of features used in QFI and QFJ. To

test this hypothesis, we include the features one by one in QFJ, and examine how the performance changes when more and more features are used. We cumulate the features in the order of feature importance (starting from the most important one). The feature importance is based on the learned feature weights (absolute value of the weights) we shown in Table 3.5 and Table 3.6.

Table 4.10 reports the PRF performance for QFJ each time when including an additional feature. Note that the model needs at least one item feature and one item pair feature, therefore we start with adding the most important item feature, *ListTermFreq.ListIDF*, and the most important item pair feature, *ContextListSim*. Then we add other features in the feature importance order. We report the improvement in PRF performance when adding a feature by ΔPRF. $\Delta PRF\%$ shows the improvement in percentage. We test if the changes are statistically significant based on paired t-test, and report p-values in the table. Results for QDM, pLSA and LDA are also included for comparison.

From Table 4.10, we find that when starting with the first three features, QFJ could not outperforms the three baselines (QDM, pLSA and LDA) in PRF, but as QFJ incorporates more and more features, it starts to outperform the baselines and the improvement generally increases as it incorporates more and more features. More specifically, QFJ start outperforms the three baselines after adding the fourth feature *ContentSiteFreq* with PRF 0.4159, and eventually achieves 0.4615 in PRF when using all the features. This verifies our hypothesis that the rich set of features is the reason for our models' improvements over the baselines. However, we also notice that PRF performance increases much more slowly after the first few features, suggesting that some of the features are correlated or do not provide much additional information. This implies that we could exclude those features to save some computation when efficiency is critical.

Table 4.10: PRF performance when cumulating features in QFJ. ΔPRF reports the improvement in PRF after adding each feature. $\Delta PRF\%$ shows the improvements in percentages. P-values are from paired t-tests for testing the improvements. QDM, pLSA, LDA results tuned on PRF are also included for comparison.

Cumulated feature	PRF	ΔPRF	ΔPRF%	P-value
+ListTermFreq.ListIDF	N/A	N/A	N/A	N/A
+ContextListSim	0.2789	0.2789	N/A	N/A
+ListSelectSiteFreq	0.2850	0.0061	2.19%	0.3593
+ContentSiteFreq	0.4159	0.1309	45.93%	9.9×10^{-31}
+ListTextSiteFreq	0.4339	0.0180	4.33%	0.0002
+ListSelectPageFreq	0.4358	0.0019	0.44%	0.1070
+ListTrTermFreq	0.4359	0.0001	0.02%	0.9841
+ListTdPageFreq	0.4347	-0.0012	-0.28%	0.6938
+ContentTermFreq.IDF	0.4366	0.0019	0.44%	0.2587
+ContextTextSim	0.4477	0.0111	2.54%	0.0184
+ContentWPageFreq	0.4501	0.0024	0.54%	0.5587
+ListTrSiteFreq	0.4542	0.0041	0.91%	0.1309
+ListIDF	0.4602	0.0060	1.32%	0.0786
+ListTdSiteFreq	0.4577	-0.0025	-0.54%	0.1710
+IDF	0.4574	-0.0003	-0.07%	0.9159
+ListTextPageFreq	0.4592	0.0018	0.39%	0.4030
+ContentPageFreq	0.4595	0.0003	0.07%	0.9042
+ContentTermFreq	0.4639	0.0044	0.96%	0.0509
+ListUlPageFreq	0.4647	0.0008	0.17%	0.8022
+ListSelectTermFreq	0.4634	-0.0013	-0.28%	0.5329
+ListTdTermFreq	0.4631	-0.0003	-0.06%	0.9054
+Length	0.4640	0.0009	0.19%	0.7304
+TitleTermFreq	0.4620	-0.0020	-0.43%	0.4699
+ListUlSiteFreq	0.4614	-0.0006	-0.13%	0.8257
+ListUlTermFreq	0.4606	-0.0008	-0.17%	0.7024
+ListOlSiteFreq	0.4597	-0.0009	-0.20%	0.7043
+TitleSiteFreq	0.4636	0.0039	0.85%	0.0873
+ListOlTermFreq	0.4628	-0.0008	-0.17%	0.6835
+ListOlPageFreq	0.4653	0.0025	0.54%	0.3425
+ListCooccur	0.4619	-0.0034	-0.73%	0.1610
+TitlePageFreq	0.4649	0.0030	0.65%	0.2353
+ListTrPageFreq	0.4666	0.0017	0.37%	0.5285
+ListTextTermFreq	0.4675	0.0009	0.19%	0.7565
+LengthDiff	0.4615	-0.0060	-1.28%	0.0247
QDM	0.3279	N/A	N/A	N/A
pLSA	0.3411	N/A	N/A	N/A
LDA	0.3293	N/A	N/A	N/A

In addition, Table 4.10 also suggests several effective features, including *ListTermFreq.ListIDF*, *ContextListSim*, *ContentSiteFreq*, *ListTextSiteFreq* and *ContextTextSim*. Using only the two features, *ListTermFreq.ListIDF* and *ContextListSim*, QFJ can obtain over 60% of the PRF performance it achieves when using all the 33 features. When adding *ContentSiteFreq*, *ListTextSiteFreq*, *ContextTextSim*, we observe statistically significant improvement in PRF. This observation is consistent with our previous feature analysis.

4.4.6 Comparing Extraction Patterns

Similar to the feature ablation experiments, we investigate the effectiveness of different candidate list extraction patterns (Section 3.4) by removing one extraction pattern at a time. Note this pattern ablation not only removes the candidate lists we used for refining facets, but also will affect the features we used in QFI and QFJ. For example, when excluding the SELECT pattern, the feature *ListSelectTermFreq* will also be excluded. We report the results for QFI tuned on PRF in Table 4.11. From Table 4.11, we can see that the lexical pattern plays a very importance role in

Table 4.11: PRF performance changes when suppressing each candidate list extraction pattern. ΔPRF shows the PRF performance change when excluding the corresponding extraction pattern. ΔPRF% shows the PRF change in percentage. The p-values reported are based on paired t-test. These patterns are explained in Section 3.4). The results are based on QFI model tuned on PRF (other results are similar).

Pattern	PRF	ΔPRF	ΔPRF%	p-value
Lexical	0.3927	-0.0793	-16.80%	1.5×10^{-13}
UL	0.4268	-0.0452	-9.58%	2.5×10^{-7}
SELECT	0.4378	-0.0342	-7.25%	4.8×10^{-4}
TD	0.4676	-0.0044	-0.93%	0.2628
TR	0.4705	-0.0015	-0.32%	0.7147
OL	0.4724	0.0004	0.08%	0.9009

query facet extraction. When excluding this lexical pattern, the PRF performance drop 16.80%. Other important patterns are UL and SELECT. On the contrary,

the patterns based on tables (TD, TR) and ordered list (OL) are found to be less important – excluding them does not affect the results very much.

4.5 Summary

In this chapter, we designed an intrinsic evaluation to directly evaluate generated query facets by comparing them with human created ones. We described how to collect human annotations for query facets as ground truth. We designed an evaluation measure that combines recall and precision of facet terms with grouping quality. We use this intrinsic evaluation to compare different query facet extraction models. Experimental results show that our supervised methods (QFI/QFJ), described in Chapter 3, can take advantage of a richer set of features and outperform other unsupervised methods. Our feature analysis suggests several informative features for query facet extraction, including *ContextListSim*, *ListTermFreq.ListIDF*, *ContextTextSim* and *ListTextSiteFreq*. Our analysis on the candidate extraction patterns shows that the lexical pattern, UL pattern and SELECT pattern are more important than other patterns.

One thing we note is that term precision tends to be low while term recall is relatively high in these results. In the next chapter, we explore methods for boosting precision on the assumption that it is often more important.

CHAPTER 5

PRECISION-ORIENTED QUERY FACET EXTRACTION

5.1 Introduction

While the query facet extraction approach presented in Chapter 3 and intrinsic evaluation in Chapter 4, provide a promising direction for solving the open-domain facet generation problem, it neglects the precision-oriented perspective of the task, which we believe is important in practical use. As in many precision-oriented information retrieval tasks, we believe users are likely to care more about "facet precision" than "facet recall". That is, users may care more about the correctness of presented facets (*e.g.*, are the terms in the airline facet indeed about airlines, and are the airline terms grouped together in a same facet) than the completeness of facets (*e.g.*, are all possible facets for that query presented, and are all possible airline terms included the results?). In other words, mistakes of presenting wrong terms in a facet, or grouping terms incorrectly are more severe than omitting some facets or terms in facets. The work presented in Chapter 3 and Chapter 4 does not consider this precision-oriented factor when designing query facet extraction models or evaluating extraction results. Therefore, it is unclear if these models can adapt to such scenarios.

In this chapter, we study query facet extraction under the precision-oriented scenario, and improve extraction performance under those scenarios from two perspectives.

First, we find the learning objective used in our query faceting models are not ideal for the task especially under the precision-oriented scenario. The proposed model is trained by maximum likelihood estimation on labeled training data. However, likeli-

hood can be loosely related to the performance measure under the precision-oriented scenario. In this chapter, we propose to directly maximize the performance measure $PRF_{\alpha,\beta}$ instead of likelihood during training using an empirical utility maximization (EUM) approach. However, exact optimization on the performance measure is difficult due to the non-continuous and non-differentiable nature of information retrieval measures. We address this problem by approximating the performance measure using its expectation. We show that this empirical utility maximization approach significantly improves over previous approaches under precision-oriented scenarios, suggesting utility is a better learning objective than likelihood, and our expectation-based approximation is effective.

Second, we improve extraction performance by a selective method that shows facets for good performing queries and avoids poor performing ones. We find that extraction performance varies for different queries – some queries are naturally more difficult than others for extracting query facets. In the precision-oriented scenario, it may be more desirable to avoid showing facets for those poor performing queries and leave the users with a clean keyword-search interface. A key problem, however, is how to predict the extraction performance. To solve this problem, we propose a simple and effective score based on the expectation of the performance measure. We find the score has a strong correlation with the performance measure, and when used in the selective method, it can significantly improve the average performance with fair coverage over the whole query set.

The rest of this chapter is organized as follows. In Section 5.2, we briefly review related work. In Section 5.3, we revisit the $PRF_{\alpha,\beta}$ measures used in the intrinsic evaluation (Chapter 4) in order to help develop our empirical utility maximization approach in Section 5.4, and our selective method for query faceting in Section 5.5. We carry out experiments in Section 5.6.

5.2 Related Work

5.2.1 Directly Optimizing Performance Measures

Lots of previous work has proposed to directly optimize performance measures in learning for various information retrieval tasks, including ranking (Metzler et al., 2005; Xu et al., 2008; Xu and Li, 2007; Cossock and Zhang, 2006; Quoc and Le, 2007; de Almeida et al., 2007) and classification (Musicant et al., 2003; Joachims, 2005; Jansche, 2005). While higher performance is expected by doing so, it is usually difficult due to the non-continuous and non-differentiable nature of information retrieval measures. From the perspective of the loss function optimization, existing solutions fall into three categories (Xu et al., 2008). First, one can minimize the upper bounds of the basic loss function defined on the performance measures (Xu and Li, 2007; Joachims, 2005; Yue et al., 2007). Second, one can approximate the the performance measures with functions that are easy to handle (Jansche, 2005; Cossock and Zhang, 2006). Our work belongs to this category; it approximates the performance measure using a continuous function based on its expectation. Third, one can use specially designed technologies for optimizing the non-smooth performance measures (Quoc and Le, 2007; de Almeida et al., 2007).

More related to our problem, Jansche proposed to train a logistic regression model by directly optimizing F-measures (Jansche, 2005). The work approximated integer quantities in F-measures based on their probabilities, and thus made the optimization target continuous and differentiable. Then it trained the logistic regression model by optimizing the approximated F-measures on the training data. The method is also referred to as empirical utility maximization (or empirical risk minimization) (Ye et al., 2012), which maximizes the expected utility (or performance) by its average utility on the training data as an approximation. The model and measure we study in this paper (see Section 3.6 and 4.3) is similar to Jansche's work, and we use similar

approximation strategy in order to perform direct optimization on the performance measure.

5.2.2 Performance Prediction and Selective Methods

Previous work on performance prediction in information retrieval primarily focused on the core ranking problem. Many predictors/scores have been introduced for predicting retrieval performance, such as clarity score (Cronen-Townsend et al., 2002), average IDF (Tomlinson, 2004) and robustness score (Zhou and Croft, 2006). Learning methods, such as regression models, have also been used to combine different factors for predicting retrieval performance (Kwok et al., 2004; Balasubramanian et al., 2010; Yom-Tov et al., 2005).

One application of retrieval performance prediction is to allow the systems to invoke alternative retrieval strategies for different queries according to their performance. For example, Yom-Tov et al. (2005) and Amati et al. (2004) showed that retrieval performance prediction can be used to improve the effectiveness of a search engine, by performing selective automatic query expansion for "easy" queries only. Our selective method for query facet extraction is similar to these methods in spirit – we want to selectively apply query facet extraction for "easy" queries only. However, to the best of our knowledge, no existing work has studied performance prediction for query facet extraction.

5.3 $PRF_{\alpha,\beta}$ Measure

Our empirical utility maximization approach, selective method for query faceting and evaluations for them are all based on the $PRF_{\alpha,\beta}$ measure that was introduced in Chapter 4. In this section, we revisit this measure, and reformulate it in a way that will make the development for the utility maximization approach and the selective method easier. Recall that the $PRF_{\alpha,\beta}$ measure combines term precision, term

recall, and term clustering performance. We will describe the different aspects in the following sections.

5.3.1 Notation

We continue to use notation defined in Section 3.6.1 and Section 4.3.1. The term label y_i indicates whether the list item t_i is indeed a facet term. The pair label $z_{i,j}$ indicates whether the list item t_i and t_j are in the same query facet. To help describe the measure, we use superscript "$*$" to distinguish ground truth labels from system predicted labels. For example, y_i^* is a ground truth term label, while y_i is a term label predicted by the system. Similarly, $z_{i,j}^*$ is a ground truth pair label, while $z_{i,j}$ is a predicted pair label.

5.3.2 Term Precision and Recall

In $PRF_{\alpha,\beta}$, the classification performance is measured by term precision (*i.e.*, precision of the selected candidate terms being facet terms) and term recall (*i.e.*, recall of facet terms). They can be formulated as below, where subscript "c", "s", "g" stands for "correct", "system", "ground truth" respectively.

- Term precision: $TP = \frac{T_c}{T_s}$, where T_c is the number of correct facet term selected, T_s is the number of terms select by the system.
- Term recall: $TR = \frac{T_c}{T_g}$, where T_c is as defined above, T_g is the number of facet terms in the ground truth.
- Term F1: $TF = \frac{2T_c}{T_s+T_g}$ is the F1 combination (or harmonic mean) of TP and TR.

The Quantities T_c, T_s, T_g can be more precisely defined using term labels y_i and y_i^* as

$$T_c = \sum_i y_i y_i^*, \quad T_s = \sum_i y_i, \quad T_g = \sum_i y_i^*. \tag{5.1}$$

5.3.3 Term clustering

system fails to find a facet term, by assuming it being a singleton, the clustering performance will be hurt (unless the facet term is a singleton in the ground truth). Empirically, we find systems return large sized facets when tuned on term clustering performance based on the adjusting. For example, on average, QFI returns 509.8 terms per query, while there is only 81.2 facet terms per query in the ground truth. Therefore by combining term precision, recall and clustering performance, $PRF_{\alpha,\beta}$ actually double-counts the term recall factor by this adjusting when measuring clustering performance.

In $PRF_{\alpha,\beta}$, term clustering performance is measured by pair-counting F1 measure after clustering adjusting (Section 4.3.3). Here the pair-counting F1 measure treats term clustering as classification on whether each pair of terms is in the same facet, and then combines pair precision and recall using the F1 measure. Pair precision and recall can be formulated as below. (The subscripts carry the same meaning as in term precision and recall.)

- pair precision: $PP = \frac{P_c}{P_s}$, where P_c is the number of term pairs the model clustered together that are indeed in the same facet in the ground truth, P_s is the number of term pairs the model clustered together.
- pair recall: $PR = \frac{P_c}{P_g}$, where P_c is as defined above, T_g is the number of term pairs clustered together in the ground truth.
- pair F1: $PF = \frac{2P_c}{P_s+P_g}$ is the F1 combination (or the harmonic mean) of PP and PR.

The quantities P_c, P_s, P_g can be more precisely defined using term labels y_i,y_i^* and pair labels $z_{i,j}$, $z_{i,j}^*$ as

$$P_c = \sum_{i,j} z_{i,j} z_{i,j}^*,\ P_s = \sum_{i,j} z_{i,j} y_i^* y_j^*,\ P_g = \sum_{i,j} z_{i,j}^* y_i y_j, \tag{5.2}$$

where term labels y_i,y_i^* are used to perform the cluster adjusting (Section 4.3.3), after which only the clustering performance on the correctly selected facet terms are evaluated.

5.3.4 Combining term precision, recall and clustering

The quality of query facet extraction is intrinsically multi-faceted. Different applications or scenarios might have different emphases among the term precision, recall and clustering. To address this issue, $PRF_{\alpha,\beta}$ combines the three factors together, using weighted harmonic mean. We repeat its formulation below,

$$PRF_{\alpha,\beta}(TP, TR, PF) = \frac{(\alpha^2 + \beta^2 + 1)}{\frac{\alpha^2}{TP} + \frac{\beta^2}{TR} + \frac{1}{PF}}. \tag{5.3}$$

Note that $\alpha, \beta \in [0, +\infty)$ are used to control the weight between the three factors. α and β can be interpreted as the importance of TP and TR compared to PF respectively (Section 4.3.4).

To evaluate query facet extraction under the precision-oriented scenario, we can set a high α and/or low β. For example, we can set $\alpha=2, \beta=1$ to evaluate the case where TP is twice as important as TR and PF. Perhaps more reasonably, we can only down-weight the recall factor, by setting $\alpha=1, \beta=1/3$ to evaluate the case where TP and PF are three times as important as TR.

To help develop our empirical utility maximization approach, we rewrite $PRF_{\alpha,\beta}$ as a function of term and pair quantities $T_c, T_s, T_g, P_c, P_s, P_g$ as

$$PRF_{\alpha,\beta}(T_c, T_s, T_g, P_c, P_s, P_g) = \frac{2(\alpha^2 + \beta^2 + 1)T_c T_p}{2\alpha^2 T_s P_c + 2\beta^2 T_g P_c + T_c P_s + T_c P_g}. \tag{5.4}$$

It is easy to see $PRF_{\alpha,\beta}$ can be also rewritten as a function of predicted labels and ground truth labels as $PRF_{\alpha,\beta}(Y, Z, Y^*, Z^*)$ by substituting term and pair quantities using Equation 5.1 and 5.2.

5.4 Empirical Utility Maximization

In this section, we describe our empirical utility maximization (EUM) approach that directly optimizes $PRF_{\alpha,\beta}$ for training the query faceting (QF) model (described in Section 3.6).

The QF model can be viewed as a model which takes in candidate terms and term pairs and predicts their labels, $(Y, Z) = h(T_{\mathcal{L}}, P_{\mathcal{L}}; \lambda, \mu)$. The parameters λ, μ are trained by maximizing the conditional likelihood of the labels, $l(\lambda, \mu)$ as defined in Equation 3.5. One problem with the maximum likelihood estimation is that the likelihood target can be loosely related to performance measure $PRF_{\alpha,\beta}$, especially in the precision-oriented scenario, where term recall is less important than other factors (as we will show in Section 5.6).

Therefore, we propose an alternative way of training the model $h(T_{\mathcal{F}}, P_{\mathcal{F}})$ by directly optimizing the $PRF_{\alpha,\beta}$ measure. Our goal is to maximize the expected utility (or performance),

$$E_{\mathcal{P}}\big[PRF_{\alpha,\beta}(h(T_{\mathcal{L}}, P_{\mathcal{L}}), Y^*, Z^*)\big], \tag{5.5}$$

where $\mathcal{P}$ is the underlying and unknown distribution of our data $(T_{\mathcal{L}}, P_{\mathcal{L}}, Y^*, Z^*)$. In order to train the model, empirical utility maximization (or equivalently empirical risk minimization) is usually used, which tries to maximizes the above utility objective function over empirical data, $\mathcal{D} = \{T_{\mathcal{L}}^{(i)}, P_{\mathcal{L}}^{(i)}, Y^{*(i)}, Z^{*(i)} | i = 1...n\}$. The empirical utility is given below,

$$\begin{aligned} U(\lambda, \mu) &= E_{\mathcal{D}}\big[PRF_{\alpha,\beta}(h(T_{\mathcal{L}}, P_{\mathcal{L}}), Y^*, Z^*)\big] \\ &= \frac{1}{n}\sum_{i=1}^{n} PRF_{\alpha,\beta}(T_c^{(i)}, T_s^{(i)}, T_g^{(i)}, P_c^{(i)}, P_s^{(i)}, P_g^{(i)}), \end{aligned} \tag{5.6}$$

where we use the uniform distribution over empirical data to replace the unknown distribution, and replace the $PRF_{\alpha,\beta}$ term with $PRF_{\alpha,\beta}$ calculation based on term and pair quantities $PRF_{\alpha,\beta}(T_c, T_s, T_g, P_c, P_s, P_g)$ as defined in Equation 5.4.

Our goal now is to find $(\lambda, \mu) = \text{argmax}_{\lambda,\mu} U(\lambda, \mu)$. Unfortunately, this objective is difficult to optimize. The basic quantities involved are integers, and the optimization objective is a piecewise-constant function of the parameters λ, μ. The non-smoothness is because the dependent variable y_i and $z_{i,j}$ take only discrete values $\{0, 1\}$. For example, $U(\lambda, \mu)$ contains integer quantity $T_c = \sum_i y_i y_i^*$ that counts the correct facet terms labeled. According to QFI (see Section 3.6.4), y_i is predicted as either 1 or 0 by thresholding its term probability $P(y_i = 1|t_i)$ as:

$$y_i = 1\{P(y_i = 1|t_i) > w_{min}\}, \tag{5.7}$$

where $P(y_i = 1|t_i) = \frac{1}{1+\exp\{-\sum_k \lambda_k f_k(t_i)\}}$ (defined in Equation 3.1) involves parameter λ. Thus, $U(\lambda, \mu)$ is a piecewise-constant function of λ. The same applies for μ as well.

In generally, we can approximate discrete variables by their expectation to obtain a smooth objective function (Jansche, 2005). In our case, by assuming independence between all the labels, y_i can be approximated by its expectation as,

$$\widetilde{y}_i = E[y_i] = P(y_i{=}1|t_i) = \sigma(\lambda^T f(t_i)), \tag{5.8}$$

where we use $\sigma(x) = \frac{1}{1+\exp\{-x\}}$ to denote the logistic function used in Equation 3.1, and use vector-representation for λ and feature $f(t_i)$ for convenience. Similarly, we approximate $z_{i,j}$ by its expectation assuming full independent condition as

$$\begin{aligned}\widetilde{z}_{i,j} &= E[z_{i,j}] = P(z_{i,j}=1, y_i=1, y_j=1|t_i, t_j, p_{i,j}) \\ &= P(z_{i,j}=1|p_i, y_i=1, y_j=1)P(y_i=1|t_i)P(y_j=1|t_j) \\ &= \sigma(\mu^T g(p_{i,j}))\sigma(\lambda^T f(t_i))\sigma(\lambda^T f(t_j)).\end{aligned} \tag{5.9}$$

In the same way, we can approximate term and pair quantities (*i.e.*, T_c, T_s, P_c, P_s, P_g) by their expectation. It is easy to see that, under the full independence assumption between all labels, their expectation can be obtained by substituting y_i and $z_{i,j}$ in Equation 5.1 and 5.2 with their expectation $E[y_i]$ and $E[z_{i,j}]$. For example, we can approximate $T_c \approx \widetilde{T}_c$ by

$$\widetilde{T}_c = E[T_c] = \sum_i E[y_i] y_i^* = \sum_i \sigma(\lambda^T f(t_i)) y_i^*. \tag{5.10}$$

Based on the approximated term and pair quantities, we can rewrite our optimization objective as

$$\widetilde{U}(\lambda, \mu) = \frac{1}{n} \sum_{i=1}^{n} PRF_{\alpha,\beta}(\widetilde{T}_c^{(i)}, \widetilde{T}_s^{(i)}, T_g^{(i)}, \widetilde{P}_c^{(i)}, \widetilde{P}_s^{(i)}, \widetilde{P}_g^{(i)}), \tag{5.11}$$

which can now be maximized numerically. More specially, we used gradient ascent for maximizing $\widetilde{U}(\lambda, \mu)$. The derivatives of $\widetilde{U}(\lambda, \mu)$ can be easily obtained based on the derivatives of $\widetilde{y}_i$, $\widetilde{z}_{i,j}$, as we give below,

$$\begin{aligned}\nabla_\lambda \widetilde{y}_i(\lambda) &= \sigma_i(1-\sigma_i)\lambda, \\ \nabla_\lambda \widetilde{z}_{i,j}(\lambda) &= \sigma_{i,j}\sigma_i\sigma_j(2-\sigma_i-\sigma_j)\lambda, \\ \nabla_\mu \widetilde{z}_{i,j}(\mu) &= \sigma_i\sigma_j\sigma_{i,j}(1-\sigma_{i,j})\mu,\end{aligned} \tag{5.12}$$

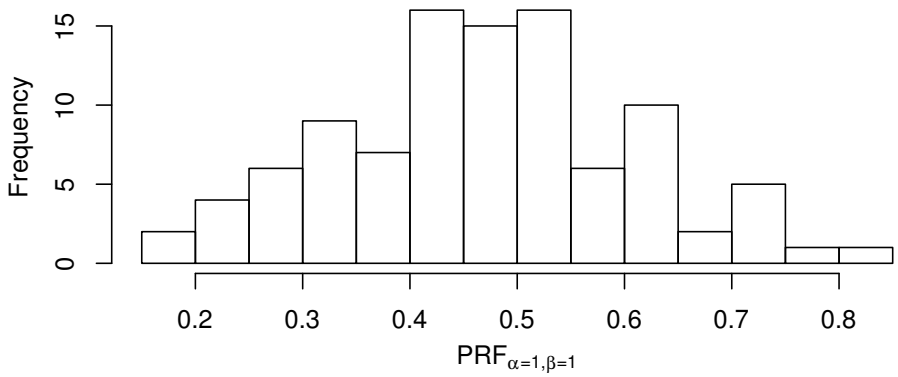
where $\sigma_i \equiv \sigma(\lambda^T f(t_i))$, $\sigma_{i,j} \equiv \sigma(\mu^T g(p_{i,j}))$. Note that the function $\widetilde{U}(\lambda, \mu)$ is generally not concave. We can deal with this problem by taking the maximum across several runs of the optimization algorithm starting from random initial values. After training,

we use the original inference QFI and QFJ as describe in Section 3.6.4 to predict labels and induce facets.

5.5 Selective Query Faceting Based on Performance Prediction

In this section we describe selective query faceting – our selective method for query facet extraction. The idea is motived by the variance in extraction performance we observed – depending on the nature of queries and extraction models, the quality of the extracted facets varies drastically from excellent to poor and complete noise. For example, queries about products, such as "toilet" and "volvo", tend to have more high-quality candidate facets extracted and are therefore easier than other complex queries, such as "self motivation", to find query facets. In Figure 5.1, we show that $PRF_{\alpha,\beta}$ could range from 0 to above 0.8 with relatively high variance. The two best performing queries (with around 0.8 $PRF_{\alpha=1,\beta=1}$) are "*used cars*" and "*bmw*".

Figure 5.1: $PRF_{\alpha=1,\beta=1}$ performance distribution. Results from QFI trained based on maximizing likelihood estimation.



5.5.1 Selective Query Faceting

Similar to the idea of selective query expansion (Cronen-Townsend et al., 2004; Amati et al., 2004; Yom-Tov et al., 2005) we can selectively present facets to users

based on the query facet extraction performance of each query. Ideally, we only show facets for good performing queries and avoid bad ones to improve user satisfaction: in the precision-oriented scenario, it may be more desirable to leave users with a clean keyword-search interface than to show poor-quality facets. To support this selective query faceting, a key problem is the prediction of the query facet extraction performance. We find a simple score based on the expectation of $PRF_{\alpha,\beta}$ can predict extraction performance fairly well.

5.5.2 Performance Prediction

In performance prediction, our goal is to predict the extraction performance for a given query with its extracted facets. We focus on predicting $PRF_{\alpha,\beta}$, and leave prediction of other measures as future work. The prediction could be done by using single indicator scores (like the clarity score in prediction retrieval performance (Cronen-Townsend et al., 2002)), or by combining different features using regression or classification models. No matter which approach, we first need to find good indicators/features for estimating the performance.

To find effective features, a natural way is to investigate the probabilistic model we have already learned in the QF method, because the learned probabilities already incorporate beliefs about the correctness of corresponding outputs. For example, we can use the term probability $P(y_i|t_i)$ defined in Equation 3.1 to estimates the chance that the output terms are indeed facet terms, and use the pairs probability $P(z_{i,j}|p_{i,j}, y_i, y_j)$ defined in Equation 3.2 to estimate the chance that the term pairs in the same extracted facets indeed belong to a query facet.

In order to use the term and pair probabilities as features, we need to aggregate them in some ways, because these probabilities are for terms and pairs, not directly for whole extracted facet set. We investigates two ways of aggregation. First, from the perspective of data fitness, we can directly use log-likelihood of extracted facets

to measure the fitness. For example, we can use the whole log-likelihood based on Equation 3.5, and we can also use the log-likelihood for only the terms or only the pairs based on the first term and second terms in the equation respectively.

Second, from the perspective of directly estimating utility (performance), we can aggregate the probabilities for estimating $PRF_{\alpha,\beta}$ directly in a similar way as our empirical utility maximization approach. More specially, we can estimate $PRF_{\alpha,\beta}$ performance based on the expected term and pair quantities under the learned model. The estimates can be obtained as follows,

$$\begin{aligned} \widehat{TP} = \frac{\sum_i P(t_i) y_i}{\sum_i y_i},\ \widehat{TR} = \frac{\sum_i P(t_i) y_i}{\sum_i P(t_i)}, \\ \widehat{PT} = \frac{\sum_{i,j} P(p_{i,j}) z_{i,j}}{\sum_{i,j} z_{i,j}},\ \widehat{PR} = \frac{\sum_{i,j} P(p_{i,j}) z_{i,j}}{\sum_{i,j} P(p_{i,j}) y_i y_j}, \end{aligned} \tag{5.13}$$

where we use $P(t_i) \equiv P(y_i = 1 | t_i)$, $P(p_{i,j}) \equiv P(z_{i,j} = 1 | p_{i,j}, y_i = 1, y_j = 1)$ for simplification. Estimates of TF, PF and can be easily obtained by substituting TP, TR, PP, PR with their estimates in the corresponding equations in Section 5.3. Estimate of $PRF_{\alpha,\beta}$ can be obtained by substituting TP, TR and PF with their estimates in Equation 4.1. We call this estimate of $PRF_{\alpha,\beta}$ the "PRF score".

To investigate the effectiveness of the two types of features, we analyze the correlation between extraction performance and each individual feature. We show the correlation results for QFI's $PRF_{\alpha=1,\beta=1}$ performance in Table 5.1. (QFI is tested using cross-validation on the QF13 dataset, which will be described in Section 5.6, under maximum likelihood estimation training. Observations are similar for other runs.)

In the table, the utility-based features TP, TR, PP, PR, TF, PF, PRF (PRF score) are estimated according to Equation 5.13 as described before. Likelihood-based feature $LL_{sum} = \sum_i \log P(y_i|t_i) + \sum_{i,j} \log P(z_{i,j}|p_{i,j}, y_i, y_j)$ calculates the likelihood of extracted facets based on Equation 3.5. tLL_{sum} and pLL_{sum} separate log-likelihood

that accounts for term and pair in LL_{sum}. Average log-likelihoods tLL_{avg}, pLL_{avg} are calculated by averaging tLL_{sum} and pLL_{sum} by the number of candidate terms $tSize$ and the number of pairs of selected terms (*i.e.*, $y_i = 1$), $pSize$.

Table 5.1: The correlation of the individual features with QFI's $PRF_{\alpha=1,\beta=1}$ performance. QFI is tested using cross-validation on QF13 dataset under maximum likelihood estimation training. Feature name abbreviation explanation: initial "t" – "term", initial "p" – "pair", LL – "log-likelihood", avg – "average", "std" – "standard deviation", $tSize$ – $|T_{\mathcal{L}}|$, $pSize$ – $|P_{\mathcal{F}}|$.

Feature	Correlation	P-value
PRF	0.6249	3.6×10^{-12}
TF	0.5933	7.7×10^{11}
TR	0.5817	2.2×10^{-10}
tLL_{avg}	0.5709	5.5×10^{-10}
$tProb_{sum}$	0.5527	2.4×10^{-9}
$tSize$	0.5512	2.8×10^{-9}
PF	0.4962	1.5×10^{-7}
PR	0.4878	2.6×10^{-7}
pLL_{sum}	-0.4513	2.4×10^{-6}
$pSize$	0.4487	2.8×10^{-6}
$pProb_{sum}$	0.4435	3.8×10^{-6}
LL_{sum}	-0.4371	5.4×10^{-6}
PP	0.4015	3.4×10^{-5}
TP	0.3336	6.9×10^{-4}
pLL_{avg}	0.3329	7.1×10^{-4}
$pProb_{std}$	-0.3162	0.0014
tLL_{sum}	-0.2317	0.0203
$pProb_{min}$	-0.1984	0.0478
$tProb_{min}$	-0.1391	0.1674
$tProb_{std}$	-0.1094	0.2787
$tProb_{max}$	0.03447	0.7335

From Table 5.1, first we find that PRF score has strong correlation (0.6249 with p-value 3.6×10^{-12}) with the performance $PRF_{\alpha=1,\beta=1}$. This suggests 1) the PRF score is a good indicator for extraction performance, and might be effective in performance prediction, and 2) our estimation of $PRF_{\alpha=1,\beta=1}$ based on its expectation is effective. Second, we find utility-based features PRF scores, $\widehat{TF}$, $\widehat{TR}$ correlate better with $PRF_{\alpha,\beta}$ performance than other likelihood-based features. This validates our

assumption that likelihood can be loosely related to the performance measure, and utility could be a better optimization objective.

We combine the proposed features in linear regression and logistic regression models. However, we find the results are not significantly better than simply using PRF score for prediction, which could be caused by the linear dependence between those features. Thus, we propose to use only the PRF score for query facet extraction performance prediction, which is simple and effective as we will show in Section 5.6. We also test other features based on statistical aggregates of the term and pair probabilities, including minimum, maximum, mean, sum and standard deviation. However, they show relatively low correlation with $PRF_{\alpha,\beta}$, and thus we do not report the results here.

After choosing PRF score as the performance predictor, selective query faceting can be easily done by thresholding this score to decide to show or avoid showing query facet results for each query. We carry out experiments to evaluate its effectiveness in next.

5.6 Experiments

Our experiments aim to investigate mainly three research questions. First, we want to test whether existing query facet extraction methods adapt to precision-oriented scenarios. Second, we want to test if our empirical utility maximization approach is effective in precision-oriented scenarios. Last, we want to test whether the PRF score can effectively predict extraction performance, and support selective query faceting. We will first describe our experimental settings, then present experimental results for each of the research questions.

5.6.1 Experimental Settings

5.6.1.1 Data

We use the same data set as in Chapter 4. We call the data set QF13. QF13 contains 100 web search queries, their top 100 web search results from a commercial web search engine, and their query facet annotation. The query facet annotation was done by pooling facet results from different models, and then having the pooled terms re-grouped by human annotators into query facets. Our candidate lists are extracted as described in Chapter 3.

5.6.1.2 Evaluation

We use $PRF_{\alpha,\beta}$ as the evaluation measures, as well as term precision (*i.e.*, *TP*), term recall (*i.e.*, *TR*), and term clustering F1 (*i.e.*, *PF*). We choose this measure because it has the flexibility of adjusting emphasis between "facet precision" and "facet recall", which naturally suits well with the precision-oriented problem. When $\alpha = \beta = 1$, $PRF_{\alpha=1,\beta=1}$ is used to evaluate the case where term precision, term recall and term clustering are equally important. To evaluate facets under precision-oriented scenarios, we set a high $\alpha \in \{2, 3, \ldots, 10\}$ with fixed $\beta = 1$. The settings correspond to the cases where term precision is twice to ten times as important as both term recall and term clustering. Without any prior knowledge, it is more fair to assume that term precision and clustering are equally important (they are both "precision" factors for query facets), therefore we will focus more on only down-weighting term recall by setting $\beta \in \{\frac{1}{2}, \frac{1}{3}, ..., \frac{1}{10}\}$ or equivalently $\frac{1}{\beta} \in \{2, 3, \ldots, 10\}$ with fixed $\alpha = 1$. These settings correspond to the case where term precision and clustering are twice to ten times as importance as term recall. As before, we evaluate top the 10 facets returned from each model.

We use 10-fold cross validation on QF13 for training, tuning (if applicable) and testing models. Models are tuned on the same $PRF_{\alpha,\beta}$ measure that they are tested

on. Unless otherwise specified, statistical significance testing is performed by using paired t-test with 0.05 as the p-value threshold.

5.6.1.3 Methods

We study five query facet extraction models briefly summarized as below.

- pLSA and LDA (Section 3.7): pLSA and LDA are applied on candidate facets for facet refining. After training, the topics are returned as query facets, by using the top terms in each topic. We tune the number of facets and number of facet terms in each facet. The topic model methods in facet refining only uses term co-occurrence information.

- QDM (Section 3.7): this is an unsupervised clustering method that applies a variation of the Quality Threshold clustering algorithm (Heyer et al., 1999) to cluster the candidate facets with bias towards important ones. This method incorporates more information than just term co-occurrence, but it is not easy to add new features into the model to further improve the performance. We tune the diameter threshold, weight threshold for valid cluster and the threshold for selecting terms.

- QFI and QFJ (Section 3.6.4): the two models incorporate a rich set of features and learn from labeled data. They were found to be the best across several evaluation measures in Chapter 4, therefore we primarily focus on them. Beyond the difference of independent and joint inference, the two models are different in that QFI has parameters that can be tuned for given measures, while QFJ does not, as it tries to optimize log-likelihoods. For QFI, we tune the weight threshold for facet terms w_{min}, and the diameter threshold dia_{max}.

We study two ways of training the graphical model (see Section 3.6) for QFI and QFJ.

- MLE (Section 3.6.3): Uses conditional likelihood as the optimization objective and performs maximum likelihood estimating for training.

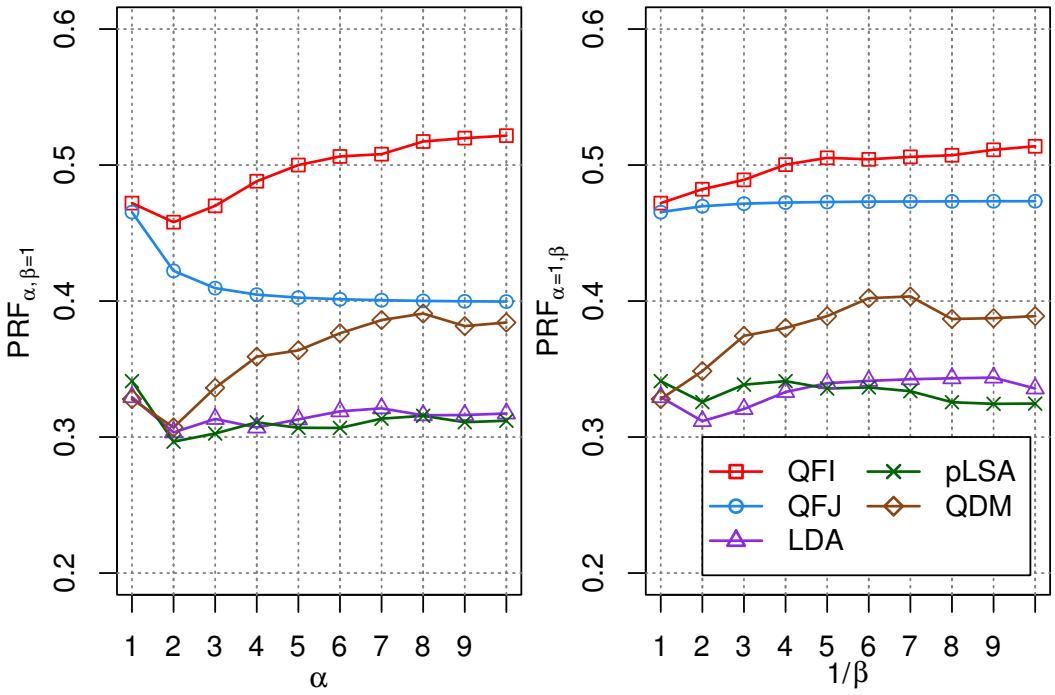
- EUM (Section 5.4): since likelihood is loosely related to the performance measure, we propose to use empirical utility maximization to directly optimize the $PRF_{\alpha,\beta}$ measure during training. We approximate $PRF_{\alpha,\beta}$ by its expectation in order to enable numerical optimization. With different α, β, we can use different versions of $PRF_{\alpha,\beta}$ as the optimization objective. We test three runs by setting $(\alpha=1, \beta=1)$, $(\alpha=2, \beta=1)$, $(\alpha=1, \beta=\frac{1}{2})$. We denote the different runs by add α, β subscript in "EUM" (*e.g.*, $EUM_{2,1}$ stands for EUM training using $PRF_{\alpha=2,\beta=1}$ as the optimization objective).

5.6.2 Existing Methods Under Precision-Oriented Scenarios

We first investigate if the five existing models can adapt to precision-oriented scenarios by evaluation based on $PRF_{\alpha,\beta}$ with different α, β settings. In Figure 5.2, we show $PRF_{\alpha,\beta}$ performance of different α (*i.e.*, term precision is more important than term recall and clustering) on the left, and of different β (*i.e.*, term precision and clustering are more important than term recall) on the right. We test all the five models with QFI, QFJ trained by MLE.

First, we find QFJ does not adapt well to precision-oriented scenarios. From the figure, we can see the superiority of QFJ over other models becomes less evident, when moving from the normal case (low α or high β) to precision-oriented cases (high α or low β). This because that QFJ tries to optimize log-likelihood for inferencing, and it cannot be tuned on the performance measures like other models. So it returns the same results for the normal case and precision-oriented scenarios. Second, we generally find that QFI and QDM can adapt better than the other models to the precision-oriented scenarios, with QFI consistently better than all the other models on both datasets. The adaptability of the two models can be explained by their

Figure 5.2: $PRF_{\alpha,\beta}$ performance with different α (left, fixed $\beta = 1$) and different β (fixed $\alpha = 1$, right) settings for existing methods on QF13.



(a) QF13, $PRF_{\alpha,\beta=1}$ (b) QF13, $PRF_{\alpha=1,\beta}$

tuning procedure. For example, depending on the target performance measure, QFI can set different threshold w_t for selecting facet terms. Overall, we find QFI (under MLE training) is the best among these existing models for the normal case, as well as precision-oriented cases.

To further analyze how QFI adapts to the precision-oriented scenarios, in Table 5.2, we report $PRF_{\alpha,\beta}$ together with TP, TR, PF and facet size (the total number of terms returned for a query) when setting different β in $PRF_{\alpha,\beta}$.

From Table 5.2, we find that as term recall factor becomes less and less important (or equivalently as the precision factors becomes more and more important), QFI becomes more and more conservative in selecting terms. The number of terms returned on average for each query ("size" in the table) decreases from 89.5 to 45.2. Term precision TP thus increases significantly, while term recall TR and term clustering

Table 5.2: $PRF_{\alpha,\beta}$ performance with its TP, TR, PF under different β settings (fixed $\alpha = 1$) for QFI on QF13. "Size" reports the average number of terms returned for each queries.

$\frac{1}{\beta}$	$PRF_{\alpha,\beta}$	TP	TR	PF	Size
1	0.4720	0.4450	0.4881	0.6209	89.5
2	0.4822	0.4896	0.4186	0.6192	70.7
3	0.4891	0.5108	0.3574	0.5989	56.5
4	0.5003	0.5291	0.3498	0.5925	53.0
5	0.5053	0.5348	0.3306	0.5928	48.9
6	0.5042	0.5343	0.3194	0.5834	47.1
7	0.5060	0.5343	0.3194	0.5834	47.1
8	0.5072	0.5343	0.3194	0.5834	47.1
9	0.5112	0.5364	0.3172	0.5864	46.7
10	0.5138	0.5365	0.3097	0.5824	45.2

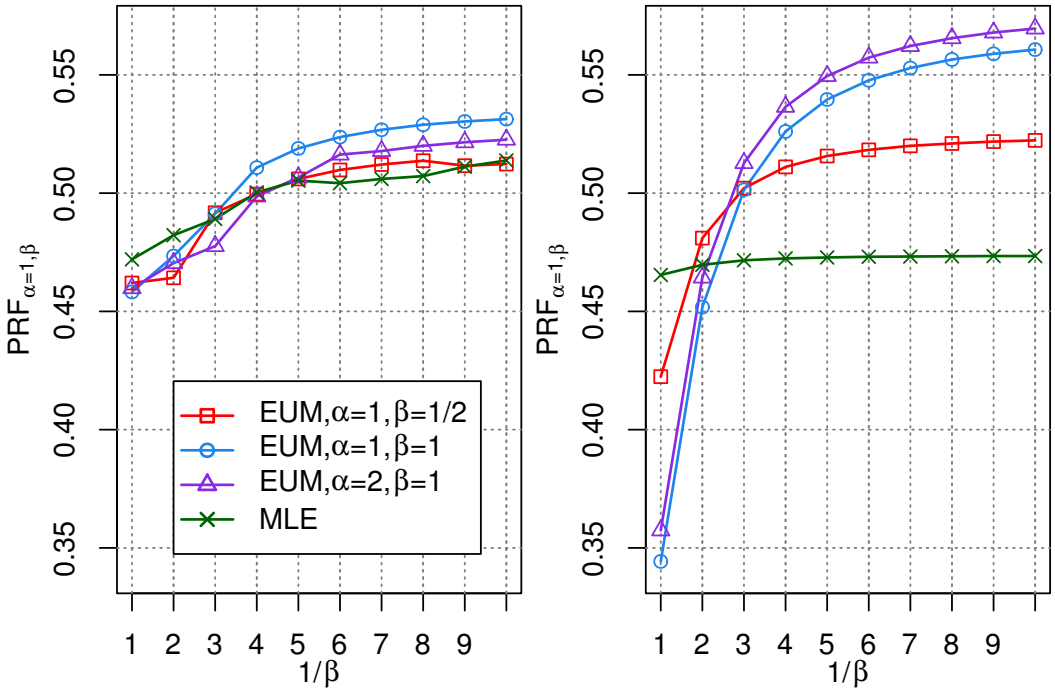
PF decrease. This indicates, by tuning on the performance measure, QFI tries to find a good balance between the tree factors for each scenarios.

5.6.3 Empirical Utility Maximization Performance

Next, we compare EUM and MLE training to test the effectiveness of the EUM approach we proposed. We first compare EUM and MLE training using both QFI and QFJ in Figure 5.3. We report results for $PRF_{\alpha=1,\beta}$ (*i.e.*, fixed $\alpha = 1$ with different β settings) on QF13. Observations are similar for other cases. The figure shows QFI with EUM and MLE training on the left, and QFJ results on the right. We report three runs of EUM, which use $PRF_{\alpha,\beta}$ under different α, β settings (specified in the legend) as the training target.

From Figure 5.3, for QFI, we find there are no statistically significant differences between MLE and EUM in most cases, even though generally EUM obtains slightly better $PRF_{\alpha,\beta}$ than MLE. This can be explained by noting that QFI under MLE has already incorporated the $PRF_{\alpha,\beta}$ learning target because it is tuned on $PRF_{\alpha,\beta}$. Essentially, we can view QFI (under MLE training) as a model that is trained on

Figure 5.3: $PRF_{\alpha=1,\beta}$ performance for MLE and EUM training using QFI (left) and QFJ (right) on QF13. The three EUM runs use $PRF_{\alpha,\beta}$ under different α, β settings (specified in the legend) as the training target.



(a) QFI (b) QFJ

likelihood to find a small tuning space to enable optimization on given performance measures by hand tuning.

Differently, for QFJ, we find EUM can improve largely over MLE under the precision-oriented scenarios. The differences between EUM and MLE are statistically significant for all $1/\beta > 2$ and for all the three EUM runs. This indicates 1) utility (performance measure) is a better optimization objective than likelihood and 2) our approximation of $PRF_{\alpha,\beta}$ based on its expectation is effective.

To study how EUM training affects QFJ in more details, as an example, we show $PRF_{\alpha=1,\beta=0.1}$ together with its TP, TR, PF and facet size in Table 5.3.

From Table 5.3, first, we can see when trained on EUM under precision-oriented settings (*i.e.*,$EUM_{2,1}$ and $EUM_{1,0.5}$), QFJ are more conservative in selecting terms

Table 5.3: $PRF_{\alpha=1,\beta=0.1}$ with TP, TR, PF for MLE and EUM training on QF13. Subscripts of EUM indicates the α, β setting used for its optimization target $PRF_{\alpha,\beta}$.

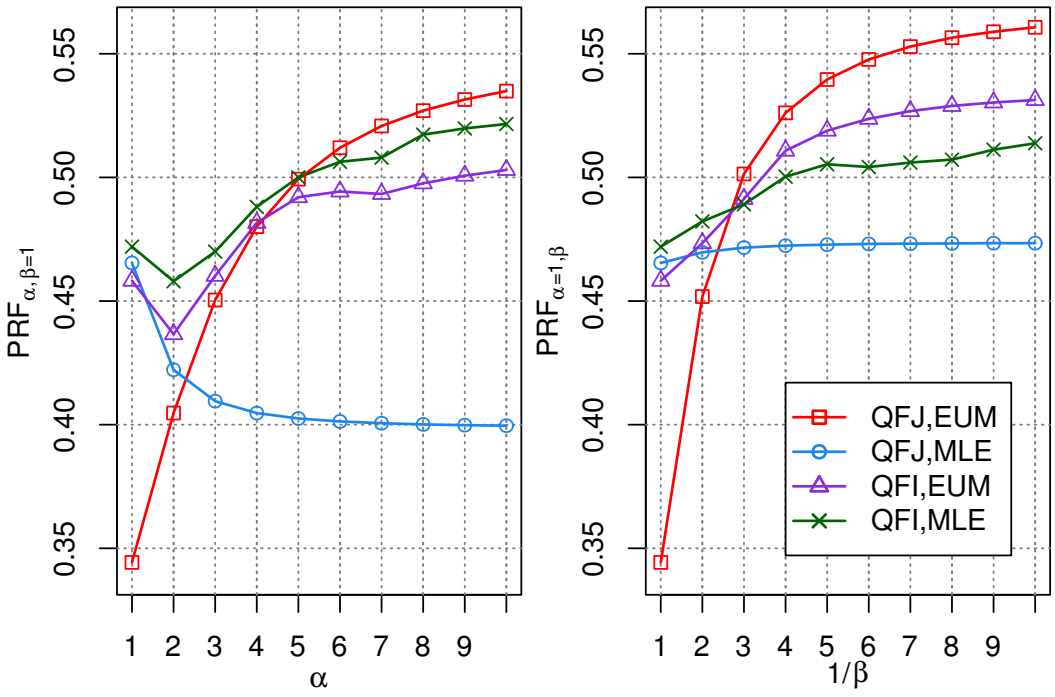
model	Training	$PRF_{\alpha,\beta}$	TP	TR	PF	Size
QFJ	MLE	0.4734	0.3986	0.4832	0.6961	97.0
QFJ	$EUM_{1,1}$	0.5223	0.4884	0.3341	0.6702	54.8
QFJ	$EUM_{2,1}$	0.5696	0.5711	0.2328	0.6705	33.9
QFJ	$EUM_{1,0.5}$	0.5607	0.5710	0.2229	0.6620	33.0

than in MLE training. When moving from MLE to EUM training, its facet size becomes much smaller (*i.e.*, 97 to 33), TP increases greatly while TR decreases substantially. This effect is desirable under the precision-oriented scenarios, in which we care much more about precision than recall, as reflected by the improvement in $PRF_{\alpha=1,\beta=0.1}$ shown in the table.

Second, by comparing $EUM_{1,1}$ with $EUM_{2,1}$, $EUM_{1,0.5}$ in Table 5.3, we can see $EUM_{2,1}$, $EUM_{1,0.5}$ trained models behave more conservatively than $EUM_{1,1}$ trained models. This suggest our training is effective – as we change the training target $PRF_{\alpha,\beta}$ parameter from $(\alpha = 1, \beta = 1)$ to $(\alpha = 2, \beta = 1)$ and $(\alpha = 1, \beta = 0.5)$), it learns that we are putting more emphasis on precision, and thus behaves more conservatively.

Last, the improvement of QFJ in precision oriented scenarios raises a question – will it outperform the previous best model, QFI, under precision-oriented scenarios? We test this in Figure 5.4. In the figure, we compare QFJ under EUM training with other baselines, including QFI under MLE (representing the state-of-the-art baseline) and EUM training. We report results under $EUM_{1,0.5}$ training on QF13 (results are similar in other cases). From the figure we find QFJ under EUM training outperforms other models in the precision-oriented scenarios. The difference between QFJ,EUM and the state-of-the-art method QFI,MLE are statistically significant for $PRF_{\alpha=1,\beta}$ when $\frac{1}{\beta} > 4$.

Figure 5.4: $PRF_{\alpha,\beta}$ performance with different α, β settings for QFI and QFJ under MLE and EUM training on QF13. $EUM_{1,0.5}$ run result is reported for EUM.



(a) $PRF_{\alpha,\beta=1}$ (b) $PRF_{\alpha=1,\beta}$

5.6.4 Extraction Performance Prediction

To predict query facet extraction performance, we build linear regression models using only the PRF score (see Section 5.5) as the feature (with intercept). We test the models for predicting $PRF_{\alpha,\beta}$ under different α, β, based on 10-fold cross validation on QF13 for QFI in Table 5.4. We report root-mean-square deviation (RMSD), Pearson correlation (R), and p-values for the significance of correlation.

The results in Table 5.4 show fairly strong RMSD values and strong positive correlations between the predicted $PRF_{\alpha,\beta}$ and real $PRF_{\alpha,\beta}$ performance for most cases. For example, p-value 1.4×10^{-11} for the first row indicates that it is extremely unlikely that the predicted $PRF_{\alpha=1,\beta=1}$ performance has no relationship with the actual performance. We also see one exception. For $PRF_{\alpha=5,\beta=1}$ we only see fair correlation,

Table 5.4: Linear regression results based on 10-fold cross-validation for predicting $PRF_{\alpha,\beta}$ performance. RMSD – root-mean-square deviation, R – Pearson correlation.

Measure	RMSD	R	p-value
$PRF_{\alpha=1,\beta=1}$	0.1110	0.6112	1.4×10^{-11}
$PRF_{\alpha=1,\beta=0.2}$	0.1800	0.5745	4.1×10^{-10}
$PRF_{\alpha=1,\beta=0.1}$	0.1882	0.5566	1.8×10^{-9}
$PRF_{\alpha=5,\beta=1}$	0.2109	0.2958	0.0028
$PRF_{\alpha=10,\beta=1}$	0.2245	0.4028	3.2×10^{-5}

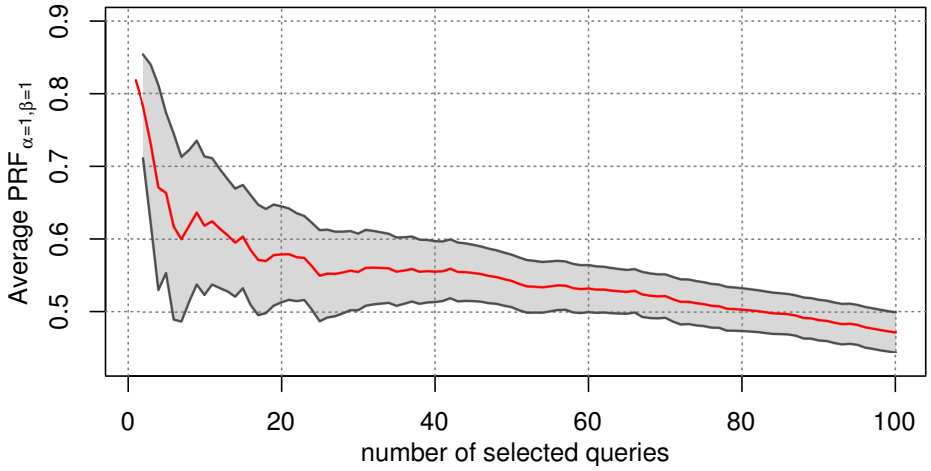
which may be because that we use $\alpha = 1, \beta = 1$ for computing our PRF score, while in $PRF_{\alpha=5,\beta=1}$ the three factors are more unbalanced weighted.

5.6.5 Evaluating Selective Query Faceting

Next, we study the effectiveness of selective query faceting based on the predicted score (from cross validation). Recall that our selective method is done by thresholding the predicted performance for deciding whether to show or avoid showing facets for each query (see Section 5.5). There is a trade-off between the performance of selected queries and the coverage of queries for query faceting. With a higher threshold, selective query faceting would select fewer queries to show facets, but users should obtain better performance for the facets that are presented to them. On the contrary, a lower threshold will result in selecting more queries to show facets, but the performance for the selected queries may be worse.

To evaluate selective query faceting, we plot the average $PRF_{\alpha,\beta}$ performance for queries selected by PRF score, when using different thresholds in Figure 5.5. The x-axis indicates the number of selected queries, while the y-axis indicates the average $PRF_{\alpha,\beta}$ performance for those selected queries. In addition to average $PRF_{\alpha,\beta}$, we also plot the standard error with 95% confidence intervals by the gray area (except for the case where only one query is selected). We report results on QF13 for a QFI run that are trained under MLE and evaluated on $PRF_{\alpha=1,\beta=1}$.

Figure 5.5: Average $PRF_{\alpha,\beta}$ performance for selected queries. The gray area indicates standard error with 95% confidence intervals. Run: $PRF_{\alpha=1,\beta=1}$ as the measure with MLE trained QFI as the extraction model



From Figure 5.5, we can see as we select fewer and fewer queries for presenting facets, generally the average performance for the selected queries increases. This indicates the query faceting method is fairly effective in selecting good performing queries and avoiding bad ones. When 20 queries are selected, we obtain 0.5792 $PRF_{\alpha=1,\beta=1}$ for the selected queries, comparing to 0.4720 when the selective method is not performed (*i.e.*, showing facets for all queries). The difference are statistically significant according to two-tailed two-sample t-test (p-value = 0.0034).

5.7 Summary

In this chapter, we studied and improved query facet extraction under precision-oriented scenarios, which could help this technique to be used practically. We find the performance expectation can be used as an approximation to directly optimize the performance measure, which significantly improves existing models under precision-oriented scenarios. We proposed a PRF score based on the expectation of $PRF_{\alpha,\beta}$ to predict extraction performance. We show this score has fairly good prediction per-

formance which enables selective query faceting that selects good performing queries to show facets, and improves the average extraction performance.

In next chapter, we will consider how to re-organize search results based on users' selection on the extracted query facets.

CHAPTER 6

FACET FEEDBACK

6.1 Introduction

Facet feedback (Kong and Allan, 2014) is to adjust (filter or re-rank) the search results based on users' selections on the provided facets (corresponding to step 5 in Figure 1.2). Boolean filtering, which filters search results by requiring selected facet terms to appear, is the dominant facet feedback method used in conventional faceted search. However, it may be too strict when extended to the open-domain setting. Boolean filtering is based on two assumptions (Zhang and Zhang, 2010): (1) users are clear about what they are looking for, and thus are able to select proper terms to restrict the results; and (2) matching between a term and a document is accurate and complete. In Faceted Web Search (FWS), that means a document that contains the selected term should be relevant to the term, and all documents relevant to that selected term should contain the term. Neither of the two assumptions are likely to hold in FWS. Therefore, we propose soft ranking models that expand original queries with user selected terms for re-ranking.

In this chapter we describe the two types of models we explore for facet feedback: Boolean filtering and soft ranking. We will first define notations used in facet feedback models in Section 6.3, and then present Boolean filtering and soft ranking models in Section 6.4 and Section 6.5 respectively. We will explore the effectiveness of these approaches in Chapter 7.

6.2 Related Work

There is a long history of using user explicit feedback to improve retrieval performance. In relevance feedback (Rocchio, 1971; Salton, 1997), documents are presented to users for judgment, after which terms are extracted from the judged relevant document, and added into the retrieval model. In the case where true relevance judgment is unavailable, the top documents are assumed to be relevant, which is called pseudo relevance feedback (Buckley et al., 1995; Abdul-Jaleel et al., 2004). Because a document is a large text unit, which can be difficult for users to judge and for the system to infer relevance information, previous work also studied user feedback on passages (Allan, 1995; Xu and Croft, 1996) and terms (Koenemann and Belkin, 1996; Tan et al., 2007).

For faceted search, Zhang and Zhang (2010) study user feedback on facets, using both Boolean filtering and soft ranking models. However, the study is based on corpora with human-created facet metadata, which is difficult to obtain for the general web. One other difference between our work and most other user feedback work is that facet feedback in our work is used to improve ranking with respect to the query subtopic specified by the feedback terms, instead of the query topic represented by the original query. This presents the scenario in FWS, where users start with a less-specified query, and then use facets to help clarify and search for subtopic information.

6.3 Notations

We use t^u to denote a **feedback term** selected by a user u, $F^u = \{t^u\}$ to denote a facet that contains feedback terms (a **feedback facet**), and $\mathcal{F}^u = \{F^u\}$ to denote the set of feedback facets. Given those, a feedback model can be formally denoted as $S'(D, Q, \mathcal{F}^u)$, which gives a score for document D according to the original query Q and the user's feedback $\mathcal{F}^u$.

6.4 Boolean Filtering Model

The Boolean model filters documents based on Boolean operations using the feedback $\mathcal{F}^u$. Similar to Zhang and Zhang (2010), we study three different Boolean conditions for filtering. We use the AND condition to require that the document contains *all* of the feedback terms in $\mathcal{F}^u$. The AND condition might be too strict, so a relaxed alternative is to use the OR condition, which requires that the document contains *at least one* of the feedback terms. The last Boolean condition, A+O, is somewhere in between the two conditions above. It use AND across different feedback facets in $\mathcal{F}^u$, and OR for terms t^u inside each facet F^u. The Boolean feedback model scores a document by

$$S'_B(D, Q, \mathcal{F}^u) = \begin{cases} S(D, Q) & \text{if } D \text{ satisfies condition } B(\mathcal{F}^u) \\ -\infty & \text{otherwise} \end{cases} \tag{6.1}$$

where condition B can be either AND, OR, or A+O, and $S(D, Q)$ is the score returned by the original retrieval model. Notice that when there is only a single feedback term, the three conditions will be equivalent; when there is only one feedback query facet (group of feedback terms), OR and A+O will be equivalent.

6.5 Soft Ranking Model

While Boolean filter models are commonly used in faceted search, it may be too strict for FWS, as explained in Section 6.1. Therefore, we also use soft ranking models, which expand the original query with feedback terms, using a linear combination as follows

$$S'_E(D, Q, \mathcal{F}^u) = \lambda S(D, Q) + (1 - \lambda) S_E(D, \mathcal{F}^u) \tag{6.2}$$

where $S(D, Q)$ is the score from the original retrieval model as before, and $S_E(D, \mathcal{F}^u)$ is the expansion part which captures the relevance between the document D and

feedback facet $\mathcal{F}^u$, using expansion model E. λ is a parameter for adjusting the weight between the two parts.

Notice that the soft ranking models are different from the extended Boolean model (Salton et al., 1983), which is sometimes referred as "soft Boolean" model (Kwok et al., 1993). The extended Boolean model or soft Boolean model was developed for Boolean queries. It retains the Boolean query structure but provides weights to both the terms and Boolean operators in the query, so that the model can soften the Boolean logic and provide ranking capacity. Instead, our soft ranking models are not for Boolean queries. They are query expansion models for ranking documents. We call them *soft ranking* models instead of *ranking* models to: 1) emphasize their difference to the *strict* Boolean filtering models described above; and 2) differentiate them with ranking models that do not have explicit user feedback.

We use two expansion models for $S_E(D, \mathcal{F}^u)$, a term and a facet expansion model. The term expansion model, ST, assigns equal weight for all the feedback terms, as follow,

$$S_{ST}(D, \mathcal{F}^u) = \frac{1}{N} \sum_{F^u \in \mathcal{F}^u} \sum_{t^u \in F^u} S(D, t^u) \tag{6.3}$$

where N is the total number of facet terms. $S(D, t^u)$ can be the original retrieval model used for the query or a different model.

The facet expansion model, SF, uses the facet structure information. It assigns equal weights between each feedback facet, and equal weight between feedback terms within the same facet, as shown below.

$$S_{SF}(D, \mathcal{F}^u) = \frac{1}{|\mathcal{F}^u|} \sum_{F^u \in \mathcal{F}^u} \frac{1}{|F^u|} \sum_{t^u \in F^u} S(D, t^u) \tag{6.4}$$

Notice that the two expansion models will be equivalent when there is only a single feedback term or when there is only one feedback facet. In our experiments, we use the Sequential Dependence Model (SDM) (Metzler and Croft, 2005) as the

baseline retrieval model $S(D, Q)$, which incorporates word unigrams, adjacent word bigrams, and adjacent word proximity. We choose SDM because it was found to be more effective than more commonly used bag-of-words models (Huston and Croft, 2014). For $S(D, t^u)$, we use the Query Likelihood model with Dirichlet smoothing as below,

$$S(D, t^u) = \sum_{w \in t^u} \log \frac{tf(w, D) + \mu \frac{tf(w, \mathcal{C})}{|\mathcal{C}|}}{|D| + \mu} \tag{6.5}$$

where w is a word in t^u, $tf(w, D)$ and $tf(w, \mathcal{C})$ are the number of occurrences of w in the document and the collection respectively; μ is the Dirichlet smoothing parameter; $|D|$ is the number of word in $|D|$, and $|\mathcal{C}|$ is the total number of words in the collection.

6.6 Summary

In this chapter, we described three Boolean filtering models (AND, OR, A+O) and proposed two soft ranking models (ST and SF) for facet feedback. In the next chapter, we develop an extrinsic evaluation method for evaluating these feedback models.

CHAPTER 7

EXTRINSIC EVALUATION

7.1 Introduction

The intrinsic evaluation proposed in Chapter 4 is not based on any particular search task, and thus may not reflect the real utility of the generated facets in assisting search. Therefore, we describe an extrinsic evaluation method which evaluates a system based on an interactive search task that incorporates Faceted Web Search (FWS). We believe the task is similar to a real application of FWS as illustrated in Figure 1.2: a user searches using an under-specified query, the FWS system provides query facets from which the user can select feedback terms that would help further specify the query, after which the FWS system uses the feedback terms for re-ranking documents.

For the extrinsic evaluation, ideally we could ask real users or carry out user studies to try each FWS systems, and measure the gain and cost for using them. The gain can be measured by the improvement of the re-ranked results using standard IR metrics like MAP or nDCG. The cost can be measured by the time spent by the users giving facet feedback. However this evaluation is difficult and expensive to extend for evaluating new systems rapidly.

We instead propose to *simulate* the user feedback process based on a user interaction model, using oracle feedback terms and facet terms collected from annotators. Both the oracle feedback and annotator feedback incrementally select all feedback terms that a user may select, which will then be used in simulation based on the user model to determine which subset of the oracle or annotator feedback terms are

selected by a user and how much time is spent giving that feedback. Finally, the systems are evaluated by the re-ranking performance together with the estimated time cost.

For the simulated FWS task, we use the TREC Web track dataset of the diversification task (Clarke et al., 2009a,b,c,d). It includes query topics that are structured as a representative set of subtopics, each related to a different user need, with relevance judgment made at the subtopic level. In our task, each subtopic is regarded as the search intent of a user, and the corresponding topic title is used as the under-specified query issued to the FWS system. For example, for query number 10 in the TREC 2009 Web Track, the title "cheap internet" is used as the initial query, and its subtopic "I want to find cheap DSL providers" is regarded as the search intent of the user.

We conduct extrinsic evaluations on combinations of different query facet generation models and facet feedback models in our experiments. We show that by using facet feedback from users, Faceted Web Search is able to assist the search task and significantly improve ranking performance. Comparing intrinsic evaluation and extrinsic evaluation on different facet generation models, we find that the intrinsic evaluation does not always reflect system utility in real application. Comparing different facet feedback models, we find that the Boolean filtering models, which are widely used in conventional faceted search, are too strict in Faceted Web Search, and less effective than soft ranking models.

In the rest of this Chapter, we first describe how we collect simulated facet feedback in Section 7.2. Then, we describe our user model for estimating the feedback time cost to users in Section 7.3. Last, we present our experiments for conducting extrinsic evaluation on different faceted web search systems (different combinations of query facet generation models and facet feedback models) in Section 7.4, followed by conclusions in Section 7.5.

7.2 Oracle and Annotator Feedback

Oracle feedback presents an idealized case of facet feedback, in which only *effective* terms – those that improve the quality of the ranked list – are selected as feedback terms. We extract oracle feedback terms by testing each single term in the presented facets. Each single candidate term is used by a facet feedback model to re-rank the documents and the candidate term is selected for the oracle if the improvement of the re-ranked documents meets a threshold. In our experiment, we use MAP as the metric and set the threshold to be 0.01. Since we have two types of feedback models, there are two sets of oracle feedback terms – one uses the Boolean filter models, and one uses the soft ranking models.

Oracle feedback is cheap to obtain for any facet system (assuming document relevance judgments are available), however it may be quite different from what actual users may select in a real interaction. Therefore, we also collect feedback terms from annotators. Our annotation interface is shown in Figure 7.1. The facet feedback annotation is done by presenting the facets (corresponding to list 1 to 5 in Figure 7.1) to an annotator with description of the information need (corresponding to subtopic description in Figure 7.1) and the initial under-specified query. The annotator is asked to select all the terms from the facets that would help address the information need (corresponding to step 2 in Figure 7.1). Ideally, we could present all the facets generated from different FWS systems for this annotation, but it would be quite expensive. In our experiment, we only present the annotator with facets collected from the intrinsic evaluation. This assumes all other facet terms generated by systems are uninteresting to the user or at least not easy for the user to select.

7.3 User Model

The user model describes how a user selects feedback terms from facets, based on which we can estimate the time cost for the user. While any reasonable user model

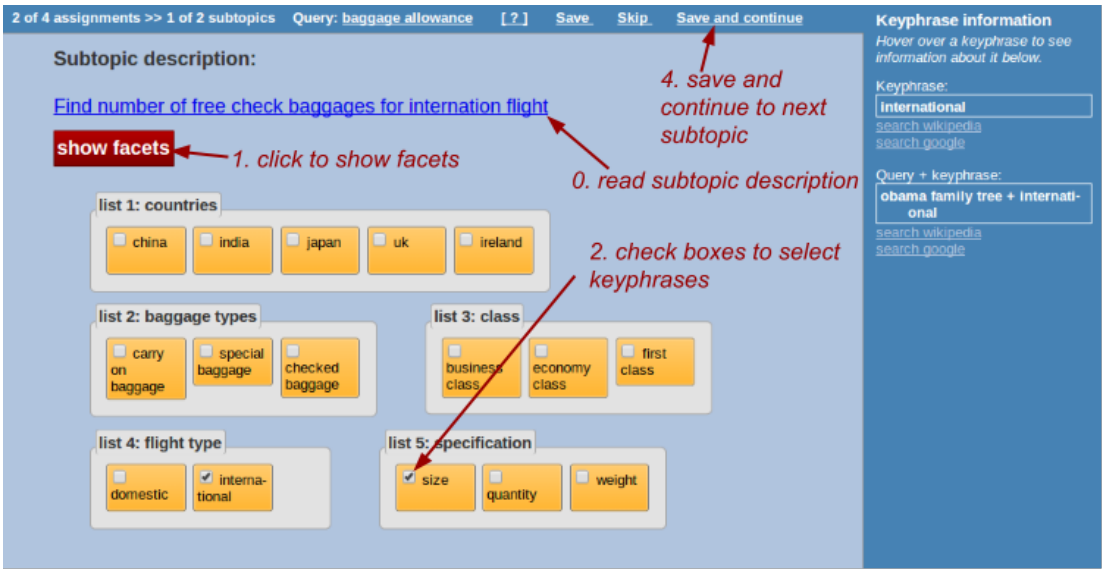


Figure 7.1: Annotation interface for facet feedback annotation.

can be brought to play here, we use a simple one, similar to the user model others have used for selecting suggestions from clusters of query auto-completions (Jain and Mishne, 2010).

Our user model is based on the structural property of facets. By grouping terms into facets, the facet interface essentially provides a skip list of these facet terms for users. More specifically, in the model, a user sequentially scans presented query facets and skips an entire facet if the user finds the facet irrelevant. Otherwise, the user will scan within the facet, sequentially reading and selecting desired facet terms, until the user finds the desired one or ones. Based on this user model, the time cost for giving facet feedback can be calculated as as,

$$T(\mathcal{F}^u) = \sum_{F^u \in \mathcal{F}^u} \left(T_f(F^u) + \sum_{t \in ts(F^u)} T_t(t) \right) \tag{7.1}$$

The righthand side of the equation contains two parts. The first part $T_f(F^u)$ is the time for scanning a facet and deciding relevance, and the second part is the time for scanning/selecting terms in the relevant facets. $ts(F^u)$ is the set of terms

scanned/selected in F^u's corresponding query facet, and $T_t(t)$ is the time used for scanning/selecting a term. Since we assume users sequentially scan the terms inside a facet, $ts(F^u)$ will include all the beginning terms in F^u's corresponding facet until the last selected term. This is based on the assumption that users are clear about what terms to select, and stop scanning after finding all of them.

To simplify the estimation, we further assume time costs are equal for scanning different facets, and equal for scanning/selecting different terms. Then the estimation becomes

$$T(\mathcal{F}^u) = |\mathcal{F}^u| \cdot T_f + |ts(\mathcal{F}^u)| \cdot T_t \tag{7.2}$$

where $|ts(\mathcal{F}^u)| = \sum_{F^u \in \mathcal{F}^u} |ts(F^u)|$ is the total number of term scanned/selected. T_f and T_t are now parameters representing the time for scanning a facet and time for scanning/selecting a term respectively.

To estimate parameters T_f and T_t, we tracked annotator behavior during the feedback annotation described in Section 7.2, including selecting / un-selecting terms and starting / exiting an annotation session. We only used annotation sessions which did not contain any un-selecting actions, and filtered out some inappropriate sessions, e.g. the annotator dwells for a long time with no activity. This selection results in 274 annotator sessions. We then extracted $|\mathcal{F}^u|$ and $|ts(\mathcal{F}^u)|$ as well as the time cost $T(\mathcal{F}^u)$ for each session, and used linear regression to fit the model to the data. When using sessions from all annotators, T_f and T_t are estimated as 2.60 and 1.60 seconds respectively, with $R^2 = 0.089$. The low R^2 is partly due to the variance introduced by using sessions of different annotators. When using one single annotator we obtain a better fit with $R^2 = 0.555$, and $T_f = 1.51$, $T_t = 0.66$, for one of the annotators. Since the estimation for T_f is about twice of T_t, for simplicity, in our experiment, we set $T_f = 2 \cdot T_t$, and report the time cost in the time unit of reading/scanning a single term.

Based on this user model, given oracle/annotator feedback, which represents all the terms that a user may select, the extrinsic evaluation works as follows. We incrementally include each term in oracle/annotator feedback as a feedback terms, and measure how ranking performance changes together with the time cost estimated based on the user model.

7.4 Experiments

7.4.1 Experiment Settings

Data set. For the document corpus, we use the ClueWeb09 Category-B collection and apply spam filtering with a threshold of 60 using the Waterloo spam scores (Cormack et al., 2011). The spam-filtered collection is stemmed using the Krovetz stemmer (Krovetz, 1993). For the query topics and subtopics, we used those from TREC Web Track's diversity task from 2009 to 2012, which also contain relevance judgments for documents with respect to each subtopic. We constrain the subtopics to have at least one relevant document in the spam-filtered collection, and this results in 196 queries and 678 query subtopics in our experiment set. For the relevance judgment, any documents that are not in the spam-filtered collection are discarded.

Annotation. We collected facet annotations as described in Section 4.2 for all 196 queries. Facets are pooled from the top 10 facets generated by runs from QDM, pLSA, LDA, QFI and QFJ. Then annotators are asked to group the terms in the pool into query facets, and to give a rating for the query facet using a scale of good (2) or fair (1). Facet annotation statistics for the good and fair facets, as well as the pooled facet, are given in Table 7.1. The table shows the average number of facet terms per query, average number of query facets per query, and average number of facet terms per facet, for each categories (fair, good, and pooled facets).

Table 7.1: Facet annotation statistics

	fair	good	pooled
#terms per query	15.8	26.5	240.0
#facets per query	2.3	3.8	40.9
#terms per facet	6.8	6.9	5.9

For the extrinsic evaluation, we also collected facet feedback annotations as described in Section 7.1 for all 678 subtopics. The statistics are given in Table 7.2, which also includes statistics for oracle feedback. The table shows the number of feedback terms selected per subtopic and the number of feedback facets per subtopic. For some subtopics, there may be no feedback terms selected, so we also report feedback coverage over subtopics in the table.

Table 7.2: Oracle and annotator feedback statistics. oracle-b and oracle-s are oracle feedback based on the Boolean filter model and soft ranking model respectively.

	annotator	oracle-b	oracle-s
#fdbk terms/subtopic	4.10	7.83	5.24
#fdbk facet/subtopic	1.36	2.40	1.93
feedback coverage	0.80	0.74	0.72

Training/testing and parameter tuning are based on 4-fold cross validation for the same splits of the 196 queries.

Significance test is performed by using paired t-test, using 0.05 as the p-value threshold.

Facet Generation Models. We compare pLSA, LDA, QDM, QFI and QFJ. $wPRF$ ($wPRF_{\alpha,\beta}$ with α and β set to 1.0) is used as the metric for parameter tuning. For pLSA and LDA, we tune the number of facets and the number of facet terms in a facet. For QDM we tune the two parameters used in the clustering algorithm, the diameter threshold for a cluster and the weight threshold for a valid cluster, as well as the parameters they used for selecting facet terms in each facet. For QFI, we tune

the weight threshold for facet terms, and the diameter threshold. For QFJ, there are no parameters that need to be tuned.

Baseline Retrieval Models and Facet Feedback Models. We use SDM as the baseline retrieval model with 0.8, 0.15, 0.05 weights for word unigrams, adjacent word bigrams, and adjacent word proximity respectively. SDM is also used as the initial retrieval model for facet generation and facet feedback. We compare different facet feedback models to SDM, including AND, OR, A+O for the Boolean filtering models, as well as ST and SF for the soft ranking models. λ in ST/SF is set to be 0.8. Dirichlet smoothing $\mu = 1500$ is used for both SDM and ST/SF. We also used other baselines including RM3 (Abdul-Jaleel et al., 2004; Lavrenko and Croft, 2001), a pseudo relevance feedback model, tuned on MAP, and xQuAD (Santos et al., 2010), a diversification model, tuned on α-NDCG (Clarke et al., 2008).

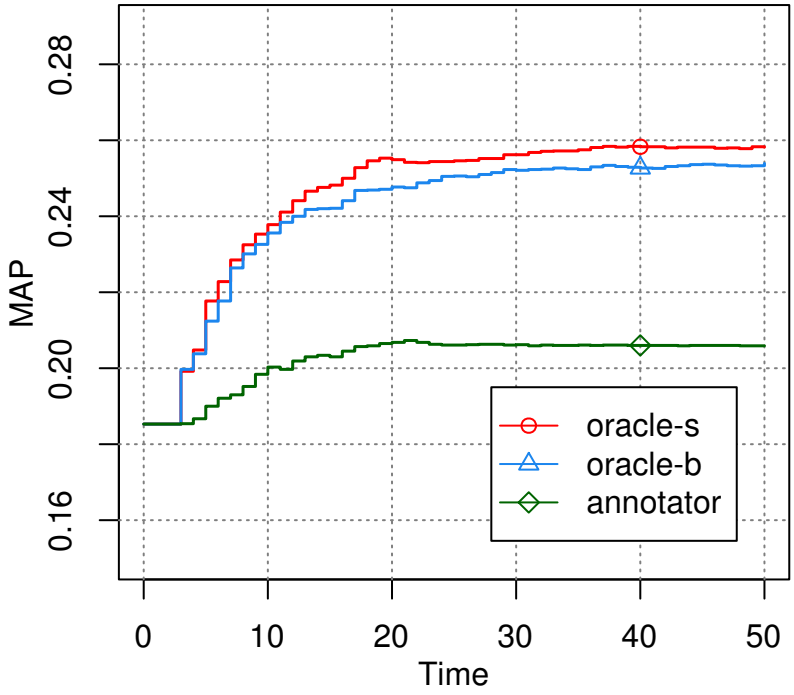
7.4.2 Oracle and Annotator Feedback

In Figure 7.2, we compare the effectiveness of oracle and annotator feedback. It shows how ranking performance changes as time cost increases, when incrementally including terms from the two types of feedback as feedback terms. The time cost is estimated by the user model described in Section 7.3. MAP is calculated with respect to the subtopic level relevance, since we are evaluating the case where the user is looking for the subtopic information. MAP value is averaged by macro-averaging – averaging for subtopics within the same query first, and then across all the queries. [1] When time is zero, no feedback terms are used, which is then just the result for the initial ranking from SDM.

In Figure 7.2, MAP increases from the SDM baseline result for both oracle and facet feedback, with the oracle ones shown to be far more effective. This shows that annotators are able to identify some useful feedback terms, but are not as effective

[1] We also measured micro-averaging, but the results are similar.

Figure 7.2: MAP change over time for oracle and annotator feedback, based on annotator facets and SF feedback model. oracle-s and oracle-b are the oracle feedback based on the Boolean filtering model and soft ranking model respectively.



as the ideal case: it seems people have a hard time knowing which terms are most likely to be successful. We further compare the feedback terms selected in oracle and annotator feedback in Table 7.3, which also supports this claim.

Table 7.3: Comparing feedback terms in annotator feedback and oracle feedback, using oracle-s as ground truth. This table shows that the annotator selects only 44% of the effective terms and that only 28% of the selected terms are effective.

Precision	Recall	F1
0.2817	0.4412	0.2179

Table 7.3 shows the overlap between oracle and annotator feedback is low according to F1. However, annotators are able to find almost half of the oracle feedback terms. Other oracle feedback terms are difficult for annotators (or users) to recognize, due to lack of background knowledge, or underlying statistical dependencies between

words that are difficult to capture. For example, for the query subtopic, "find the TIME magazine photo essay Barack Obama's Family Tree", some names of family members are selected in oracle feedback, but not by the annotator. This is because the annotator is not able to capture the relevant relationship between the names of family members and the photo essay, or simply because the annotator does not know those family members' names.

7.4.3 Comparing Facet Generation Models

7.4.3.1 Intrinsic Evaluation

To compare intrinsic and extrinsic evaluation, we also report intrinsic evaluation on different facet generation models in Table 7.4. The table shows QFI and QFJ outperform other models on the overall measure, wPRF. QFI wins because of high recall of facet terms and high F1 of facet term clustering. For rp-nDCG, QFJ and QDM are more effective. These results are consistent with our previous results in Section 4.4.

Table 7.4: Intrinsic evaluation of facet generation models.

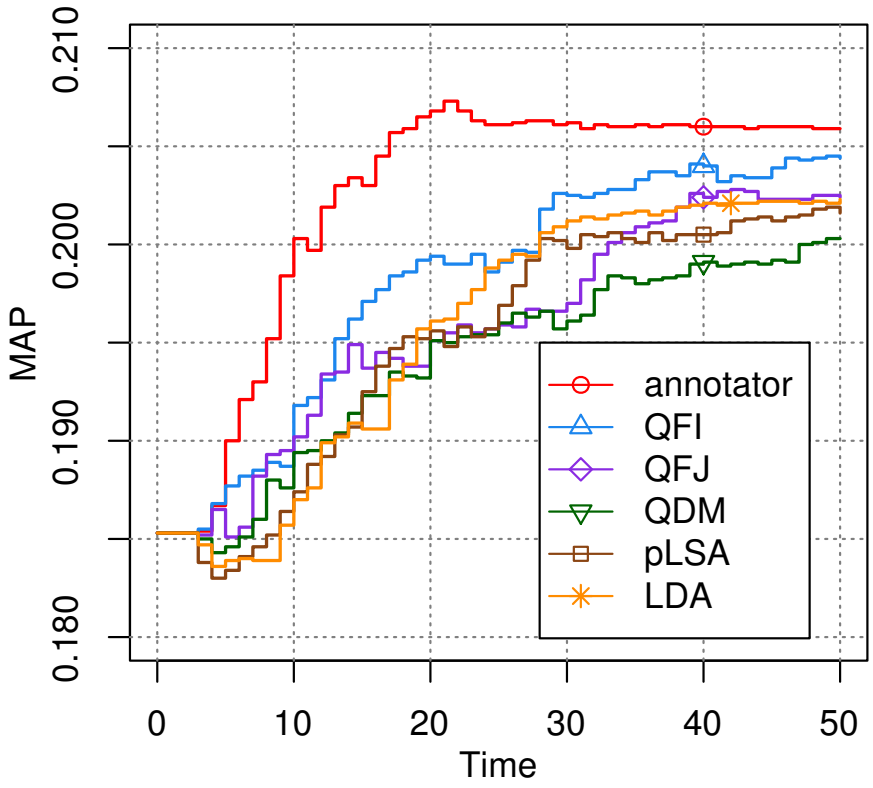
Model	wTP	wTR	wPF	wPRF	rp-nDCG
pLSA	0.2198	0.6273	0.2541	0.2521	0.0561
LDA	0.2720	0.5578	0.2345	0.2571	0.0476
QDM	0.3253	0.4024	0.2492	0.2688	0.0908
QFJ	0.3525	0.4060	0.2779	0.2836	0.1359
QFI	0.2729	0.7363	0.3859	0.3448	0.0825

7.4.3.2 Extrinsic Evaluation

Intrinsic evaluation may not reflect the utility of facets in assisting search. In Figure 7.3 we evaluate different facet generation models using extrinsic evaluation, by showing how MAP changes as time cost increases, similar to Figure 7.2.

First, Figure 7.3 shows all models are able to improve ranking from the baseline, which testifies to the potential of FWS. However, the automatically generated facets

Figure 7.3: MAP change over time for different facets generation models, based on annotator feedback and SF feedback model.



are less effective than annotator facets. MAP for annotator facets reaches 0.2 by 10 time units, while the models need much more time, ranging from 27 to 47. Second, QFI is more effective than other models over the entire time span. This is consistent with the intrinsic evaluation. Third, the comparison results for other models are less clear. QFJ and QDM are better than pLSA and LDA before 20 time units, but MAP for pLSA and LDA increases much faster afterwards, and ends at a value similar to QFJ. Comparing these results with Table 7.4, we find intrinsic metrics do not always reflect utility based on extrinsic evaluation, though the generally better performance of QFI is clear in both.

Another way to compare is to see how many terms in the presented facets are selected by annotators, as shown in Figure 7.4. The figure shows that with annotator facets a (simulated) user needs less time for selecting feedback terms. All the other

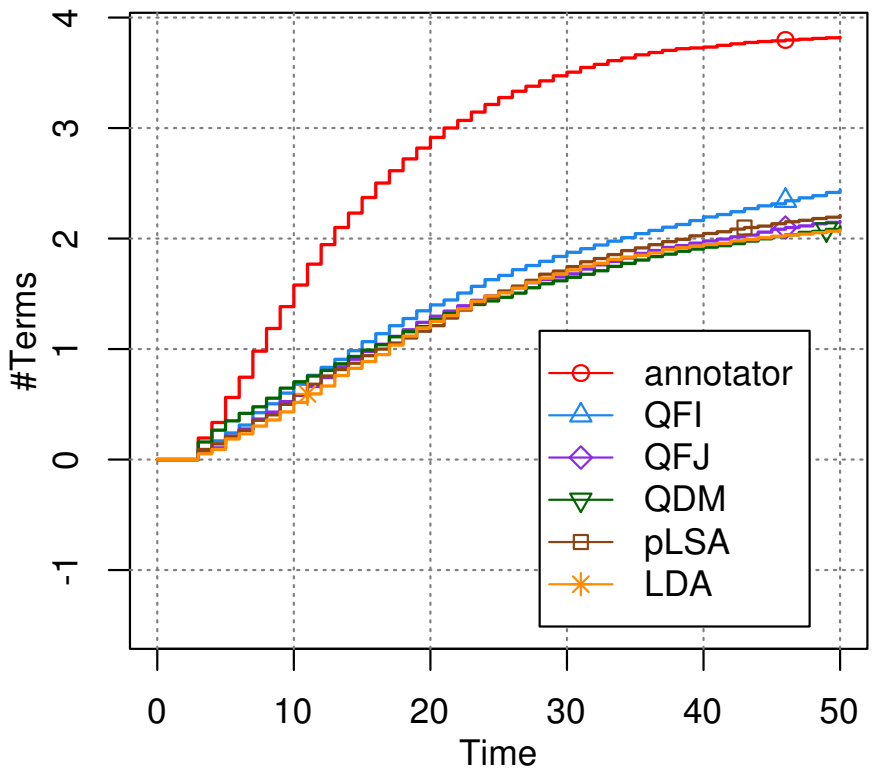


Figure 7.4: Number of feedback terms selected over time on facets generated by different models, based on annotator feedback.

facet generation approaches are similar to each other, with QDM having slightly more feedback terms at the beginning and QFI having more for the rest. This explains why QFI is the best system run in Figure 7.4 – for the same time cost, QFI has more feedback terms selected by annotators.

If we switch to using the oracle feedback facets, the difference between different facet generation models and annotator facets are no longer that big, as shown in Figure 7.5. Annotator facets are better at the beginning, but the corresponding MAP stops growing at around 20 time units. We find this is due to there not being so many facets available in the annotator facets. The number of terms and facets presented to users will affect this evaluation. In the plot, when there is not a sufficient supply of facets at some time cost, the results from a smaller time cost are used. That is, if the user runs out of facet terms to consider, performance is stuck where it last left off.

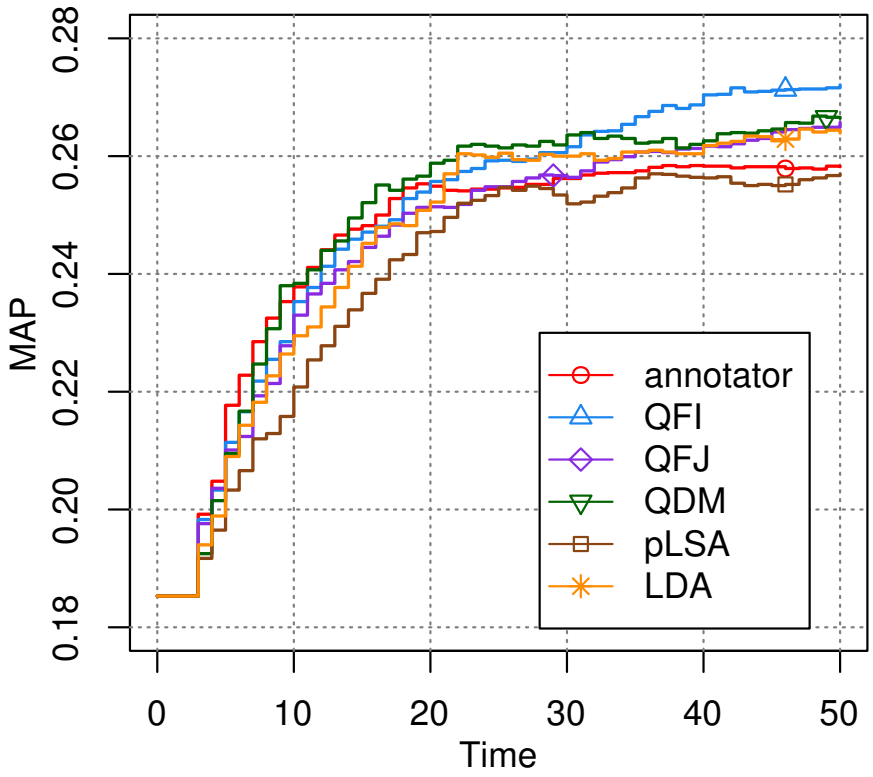
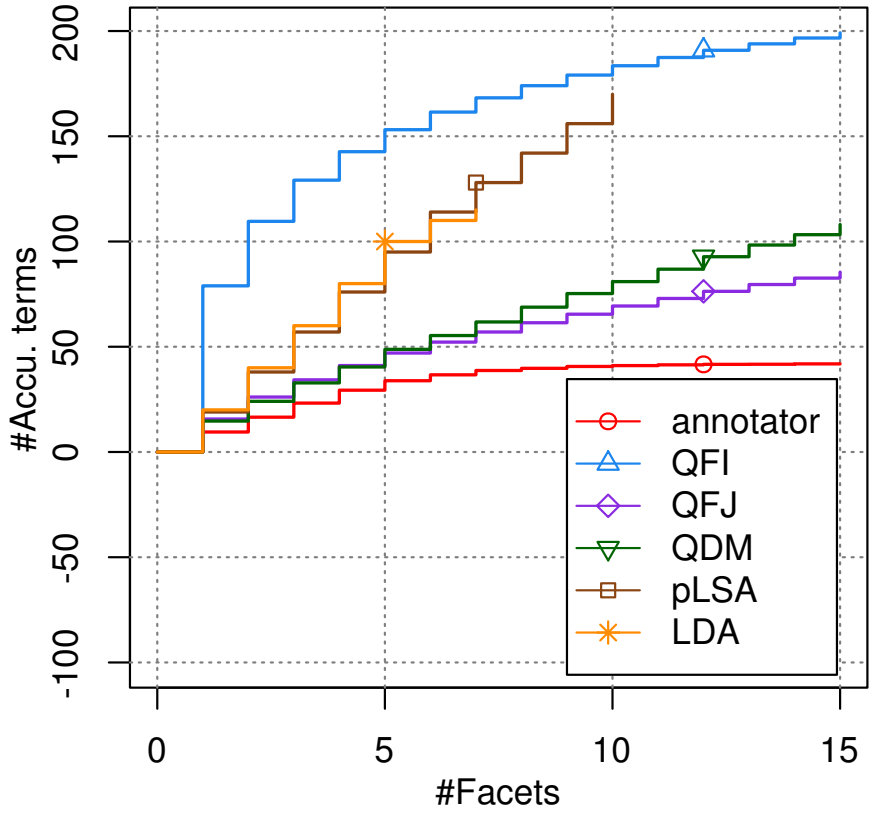


Figure 7.5: MAP change over time for different facets generation models, based on oracle feedback and SF feedback model.

To validate the comparison in Figure 7.3 and 7.5, we plot Figure 7.6 which shows the number of accumulated facet terms in the top facets generated by different models. Figure 7.6 shows all models have a sufficient supply of facet terms for these evaluations. All of them present at least 50 facet terms (on average), which will need at least 50 time units for the user to process. This obviates the concern above. However, the annotator only has on average 42.3 facet terms selected, and therefore comparison at a time larger than that might unfairly penalize the annotator facets. We also notice that the first facet in QFI is very large, and overall QFI has more terms in top facets. Since the results are tuned on wPRF with equal weight for term precision and recall, this suggests it is very likely that too much weight is assigned for recall, and a more balanced weight between wP, wR and wFP should be used in wPRF.

Figure 7.6: Accumulated number of facet terms in top facets generated from different facets generation models.



7.4.4 Comparing Facet Feedback Models

We compare different facet feedback models in Figure 7.7. It shows soft ranking models are more effective than Boolean filter models. AND is too aggressive, which hurts the ranking performance as more and more feedback terms are used. The other two Boolean filtering models, OR and A+O, are similar at the beginning. That is because in the beginning there is only one feedback facet, in which case OR and A+O will be equivalent. As more facet terms are selected, A+O performance decreases. For the two soft ranking model, SF and ST are very close, with SF slightly better as time progresses. This comparison suggests that Boolean filter models, AND and A+O, are too strict for FWS, and a soft ranking model is more effective for FWS. This situation is probably because in FWS the mapping between facet term and document

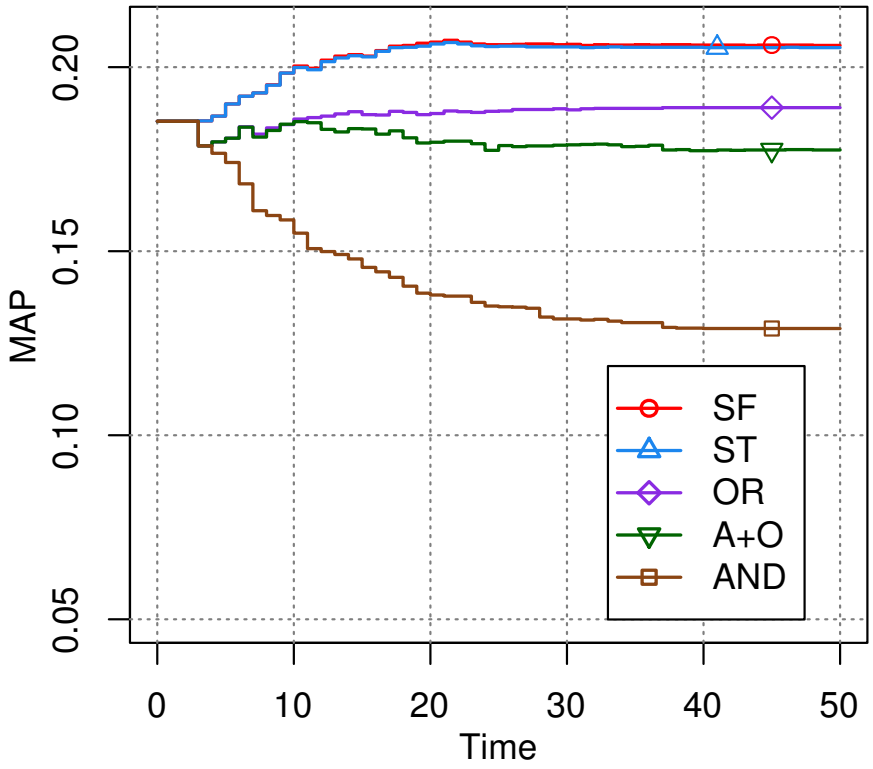


Figure 7.7: MAP change over time for different feedback models, based on annotator facets and annotator feedback.

is incomplete; a document that does not contain the exact facet term may also be relevant.

7.4.5 Comparison to Baseline Retrieval Models

We compare FWS with other baseline retrieval models in Table 7.5. QFI is used as the FWS system here, with SF as the facet feedback model. Annotator feedback terms are used, which represents a real case (not oracle) of FWS application. In the table, FWS:10 and FWS:50 are QFI runs allowed 10 and 50 time units for feedback respectively.

First, the table shows that using annotator feedback, FWS can improve ranking over the initial retrieval model, SDM. FWS also obtains better results than RM3, across all the metrics. It is also better than xQuAD for most metrics. The ob-

servations testify to the potential of FWS in assisting search. Last, when allowed more time, the results are further improved as shown by the change from FWS:10 to FWS:50.

Table 7.5: Retrieval effectiveness comparison with baselines. FWS:10 and FWS:50 are QFI runs allowed 10 and 50 time units for feedback respectively. Statistically significant differences are marked using the first letter of the retrieval model name under comparison.

Model	MAP	MRR	nDCG@10
SDM	0.1854	0.3295	0.1997
RM3	0.1886	0.3124	0.2010
xQuAD	0.1822	0.3463^{r}	0.2191
FWS:10	0.1918^{s}_{x}	$0.3476^{s,r}$	$0.2145^{s,r}$
FWS:50	$0.2044^{s,r}_{x}$	$0.3736^{s,r}$	$0.2357^{s,r}$

7.4.6 Comparison to a Retrieval Model with User Feedback

Section 7.4.5 shows that, using user feedback on facets, FWS can provide better ranking than other retrieval models that do not incorporate explicit user feedback. In this section, we want to investigate if FWS is more effective than other retrieval models that incorporate user feedback. For this, we compare FWS with a term relevance feedback model (Koenemann and Belkin, 1996; Tan et al., 2007).

The term relevance feedback model we used is based on RM3 (Abdul-Jaleel et al., 2004; Lavrenko and Croft, 2001), and thus denoted as RM3I ("I" stands for "interactive"). RM3I shows the expansion terms from RM3 (terms with high probabilities in the top ranked documents) to a user (or an annotator in our simulation). Then RM3I expands the query only with terms selected by the user in the original RM3 feedback model.

FWS and RM3I are similar in that both of them generate terms for users to select, and then use the selected terms as feedback for re-ranking. FWS and RM3I are different in two aspects. First, the terms generated can be different. FWS generates

terms based on query facet extraction. RM3I generates terms based on RM3, a pseudo relevance feedback model. Second, the presentation of terms are different. RM3I present the terms as a flat list for the user. FWS groups terms into facets (or sets of terms subsumed by an implicit label). The facet interface thus provides a skip list of these terms for users. According to our user model (Section 7.3), the user can skip an entire facet if the user finds the facet irrelevant. This potentially saves the user a lot of time in considering the terms inside irrelevant facets.

Our experiment is set up as follows. For term generation, we run RM3 for RM3I with different number of pseudo relevant documents, and report the best result, which is obtained by using top 100 documents (this is also the same setting used in FWS). For FWS, we report results for query facets generated by QFI. For user feedback, as before, we use annotator feedback terms for both models. For feedback models, FWS uses the SF facet feedback model, RM3I (as described above) uses the original RM3 model, but only with the selected terms for expansion. For both FWS and RM3I, the weights for the original query and expansion are set to 0.8 and 0.2 respectively. For estimating the time in giving user feedback, we use the user model described in Section 7.3. So time cost for FWS is estimated as before. Time cost for RM3I is estimated by assuming the list of feedback terms RM3I is presented as a single facet.

We report the results in Figure 7.8, which shows how the ranking performance changes in terms of MAP when users invest more and more time in giving feedback for FWS and RM3I. We have two observations here. First, we can see that initially (time $<$ 10 units) RM3I outperforms FWS. This indicates that, compared to FWS, RM3I could rank feedback terms that are more effective in the top of its term list. This observation implies that FWS might be improved by considering similar term ranking models/features as in RM3I. Second, we can see that when users spend more time and scan more terms/facets (time $>$ 10 units), FWS outperforms RM3I. This can be explained by the skip list structure in FWS. When users are looking for more

terms to select, the skip list structure in FWS helps users to skip irrelevant terms (in the irrelevant facets), saving them time to consider each of the irrelevant terms one by one. Thus, to achieve the same high re-ranking performance, FWS takes less time than RM3I.

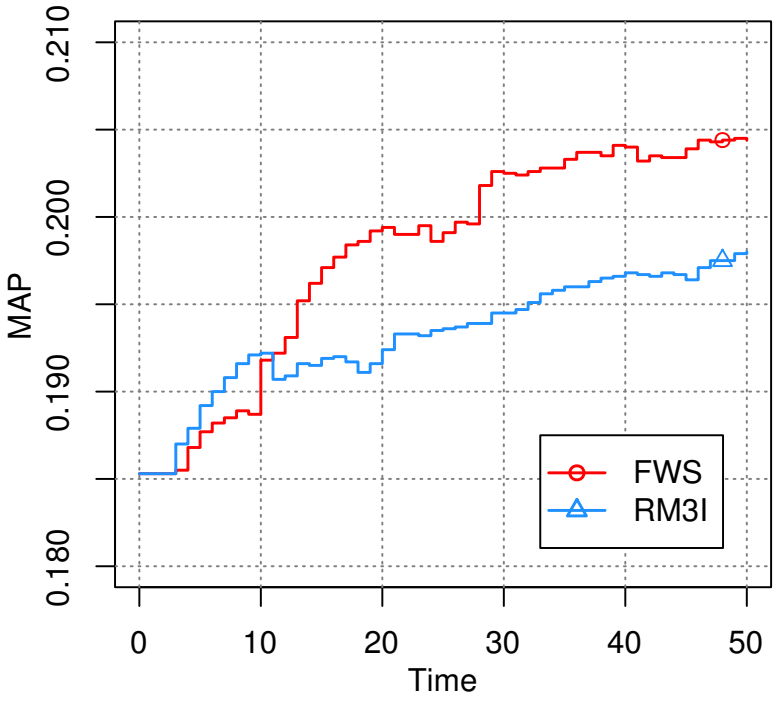


Figure 7.8: MAP change over time for FWS and RM3I (RM3 with user feedback terms). Results are based on annotator feedback terms. FWS results are based on QFI.

7.4.7 Examples

In this section, we use some system generated facets as examples, to show how FWS can assist search. We find FWS can be helpful in exploratory search. For example, for the query "cheap internet", the facets generated by QDM includes a facet of different Internet service types, {*dial up*, *dsl*, *cable*}, and a facet of different ISPs, {*netzero*, *juno*, *copper*, *toast*}. These facets can assist the user to compare different Internet service types and ISPs during his/her exploration of "cheap Internet". Another example is the query "lymphoma in dogs", in which the user may want to learn

about different aspects of lymphoma in dogs. QFJ generates the facet {*treatment*, *diagnosis*, *prognosis*, *symptoms*, ...} which represents different aspects of the query. For this query, there is a query subtopic looking for symptoms of lymphoma in dogs, which can be directly answered by another facet found by QFJ, {*vomiting*, *diarrhea*, *weight loss*, *depression*, *fever*}.

7.5 Summary

In this chapter, we developed an extrinsic evaluation method for Faceted Web Search. The extrinsic evaluation method directly measures the utility in search instead of comparing system/annotator facets as in intrinsic evaluation. We described a way to build reusable test collection for the extrinsic evaluation, and make our collected data set publicly available[2].

We also investigated different facet generation and facet feedback models based on the extrinsic evaluation method. Our experiments show, by using facet feedback from users, Faceted Web Search is able to assist the search task and significantly improve ranking performance. Our experiments also show that the skip list structure in the facet interface helps users save time in considering feedback terms in irrelevant facets. Comparing intrinsic evaluation and extrinsic evaluation on different facet generation models, we find that the intrinsic evaluation does not always reflect system utility in real application. Comparing different facet feedback models, we find that the Boolean filtering models, which are widely used in conventional faceted search, are too strict in Faceted Web Search, and less effective than soft ranking models.

[2]See http://ciir.cs.umass.edu/downloads

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this thesis, we investigated Faceted Web Search, an extension of faceted search to the open-domain web setting. We studied three fundamental problems in Faceted Web Search, namely (1) how to automatically generate facets, (2) how to re-organize search results with user feedback on facets and (3) how to evaluate generated facets and entire systems. To address these problems, we have: (1) developed query facet extraction for automatic facet generation; (2) developed an intrinsic evaluation method for evaluating generated facets; (3) developed an empirical utility maximization approach and a selective query faceting method for improving query facet extraction in precision-oriented scenarios; (4) investigated both Boolean filtering and soft ranking models for facet feedback; (5) developed an extrinsic evaluation method that evaluates entire systems in terms of their utility and cost in assisting search.

In Chapter 3, we developed query facet extraction, which extracts facets for a given query from its search results. Changing from a global approach that generates facets in advance for an entire corpus (Stoica and Hearst, 2007; Dakka and Ipeirotis, 2008) to a query-based approach, query facet extraction provides a promising direction for solving facet generation in Faceted Web Search. By focusing on the search results, query facet extraction avoids dealing with the entire web, which is large and heterogeneous. By directly generating facets for queries, it also addresses the facet recommendation problem at the same time.

For query facet extraction, we developed a supervised approach based on a graphical model to recognize query facets from the noisy candidates found. The graphical model learns how likely a candidate term is to be a facet term as well as how likely two terms are to be grouped together in a query facet, and captures the dependencies between the two factors. We proposed two algorithms (QFI and QFJ) for approximate inference on the graphical model since exact inference is intractable. Compared with other existing methods, our models can easily incorporate a rich set of features, and learn from available labeled data.

In Chapter 4, we developed an intrinsic evaluation method that evaluates generated facets by comparing them with human-created ones. We described how to collect human annotations for query facets by a pooling method. We designed $PRF_{\alpha,\beta}$, an evaluation measure that combines precision and recall of facet terms with grouping quality, using weighted harmonic mean. The measure can adjust emphasis between the three factors for different application scenarios. Experimental results based on this intrinsic evaluation show that our supervised methods (QFI and QFJ), can take advantage of a richer set of features and outperforms other unsupervised methods. Our feature analysis based on this evaluation suggests several informative features for query facet extraction, including *ContextListSim*, *ListTermFreq.ListIDF*, *ContextTextSim* and *ListTextSiteFreq*. Our analysis on the candidate extraction patterns shows that the lexical pattern, UL pattern and SELECT pattern are more important than other patterns.

In Chapter 5, we investigated query facet extraction models under precision-oriented scenarios, and improved our models in such scenarios. The precision-oriented scenarios consider a more practical setting, in which users care more about precision of presented facets than recall. From the investigation, we found that our model (QFJ), optimized based on likelihood, fails to adapt to the precision oriented scenarios, suggesting likelihood could be loosely related to the performance measure in such

scenarios. Thus, we developed an empirical utility maximization approach to optimize the performance measure instead of likelihood. However, exact optimization on the performance measure is difficult due to the non-continuous and non-differentiable nature of the objective. We solved this problem by approximating the performance measure using its expectation. Our experiments show that this empirical utility maximization approach significantly improves our query facet model (QFJ) under precision-oriented scenarios, suggesting utility is a better learning objective than likelihood, and our expectation-based approximation is effective.

Besides empirical utility maximization, we also improved query facet extraction performance in the precision-oriented scenarios by selective query faceting. In our investigation, We found query facet extraction quality varies drastically from excellent to poor and completely noisy. In the precision-oriented scenario, it may be more desirable to avoid showing facets for those poor performing queries and leave the users with a clean keyword-search interface. Thus, we proposed selective query faceting to show facets for good performing queries and avoid poor performing ones. A key problem, however, is how to predict the extraction performance. To solve the problem, we proposed a PRF score based on the expectation of $PRF_{\alpha,\beta}$ to predict the performance. We show this score has fairly good prediction performance which enables selective query faceting, and improves the performance for the selected queries with fair coverage over the entire query traffic.

In Chapter 6, we investigated both Boolean filtering and soft ranking models for facet feedback. The Boolean filtering models filter the search results based on users' selection on facets, which is the dominant feedback model in conventional faceted search. Instead, soft ranking models re-ranks the documents by expanding the original query with selected terms in facets. Our experiments (in Chapter 7) show that the Boolean filtering models are too strict in Faceted Web Search, and less effective than soft ranking models.

In Chapter 7, we developed our extrinsic evaluation method that evaluates entire Faceted Web Search systems in terms of their utility in assisting search in an interactive search task. In the search task, a user searches an under-specified query, the FWS system provides query facets from which the user can select terms in the facets that would help further specified the query, after which the FWS system uses the feedback terms for re-ranking documents. Our extrinsic evaluation considers both gain in terms of the re-ranking performance and cost in terms of the time users spend for selecting feedback terms. The re-ranking gain can be measured by standard IR metrics like MAP or nDCG. The time cost ideally can be exactly measured by carrying out user studies.

However, we noticed the user-study-based time measurement would make the evaluation difficult and expensive to extend for evaluating new systems rapidly. Thus, we proposed to simulate the user feedback process based on a user interaction model, using oracle feedback terms and feedback terms collected from human annotators. Both the oracle feedback and annotator feedback incrementally select all feedback terms that a user may select, which will then be used in simulation based on the user model to determine which subset of facet terms are selected by a user and how much time is spent giving that feedback. We also describe a way to build reusable test collection for the extrinsic evaluation, and make our collected data set publicly available[1].

Our experiments show, by using facet feedback from human annotators, Faceted Web Search is able to assist the search task and significantly improve ranking performance if allowed sufficient time for user feedback: 18.0% in NDCG@10 if we allow users to examine 50 terms in facets, and 7.4% in NDCG@10 if we allow time for examining 10 terms. Comparing FWS with a term relevance feedback model based on

[1]See http://ciir.cs.umass.edu/downloads

RM3 (Abdul-Jaleel et al., 2004; Lavrenko and Croft, 2001), we find that the skip list structure in the facet interface helps users save time in considering feedback terms in irrelevant facets, and achieve higher re-ranking performance when users are looking for more feedback terms. Comparing intrinsic evaluation and extrinsic evaluation on different facet generation models, we found that the intrinsic evaluation does not always reflect system utility in real application. Comparing different facet feedback models, as mentioned earlier, we found that the Boolean filtering models are too strict in Faceted Web Search, and less effective than soft ranking models.

8.2 Future Work

As a first extensive attempt at extending faceted search to the open-domain web, this work has some limitations, but also opens up many interesting directions for future work.

In this work, a query facet is defined as a set of coordinate terms (*e.g.*, {"*AA*", "*JetBlue*", "*Delta*"}), but with no label (*e.g.*, "*airlines*") for the set . This facet representation corresponds to one-level faceted taxonomies, in which information objects that belong to a same parent node are shown as a facet. However, explicitly showing labels for each query facets, or equivalently showing two-level faceted taxonomies, would be more desirable, as facet labels could help users quickly comprehend each query facets. There are two potential directions for solving this facet labeling problem. First, we could resort to some extraction patterns that extract facet candidates together with their labels. For example, from the sentence "*... airlines such as AA, Delta, and JetBlue.*", based on the pattern "*NP* such as *NP*, *NP*, ..., and *NP*", we can extract facet candidate {"*AA*", "*Delta*", "*JetBlue*"} together with the label "*airlines*". After candidate extraction, we need models for refining candidate facets with labels, which is also a very interesting problem. Second, we can resort to existing taxonomies. We can classify extracted query facets in to the taxonomies (*e.g.*, assign

"*AA*", "*Delta*", "*JetBlue*" as child nodes for the node "*airlines*" in the taxonomy), and then use the assigned parent nodes as labels. One problem with this direction is that existing taxonomies will almost certainly have difficulties in covering all query facets web search users are interested in.

In Chapter 3, we used a heuristic score for ranking extracted query facets. The score is defined as $score(F) = \sum_{t \in F} P(t)$, which sums up the probabilities of each term t being facet term, in order to present more facet terms in top ranks. This ranking model might be far from optimal, and there are other ranking models that could potentially improve facet ranking performance. For example, learning-to-rank models (Liu, 2009) have been well-studied, and according to their success in information retrieval, they may also work for query facet ranking. However, one problem with using learning to rank is that we need to design informative features that measures the quality of a extracted query facet.

In this work, we focused on extracting query facet from search results. However, a variety of other resources can be useful for query facet extraction. For example, existing taxonomies or knowledge bases (*e.g.*, Freebase[2]) can be useful for extracting candidate facets. Ideally, we can identify concepts (or entities) in a taxonomy that are relevant to the query, and then use the concepts with their child nodes as facet candidates (or query facets directly). For example, if we find that the concept "*airline*" is relevant to our query "*baggage allowance*", and the taxonomy contains a node for "*airline*", then we can use the concept's child nodes ("*AA*", "*Delta*", "*JetBlue*") as facet candidates. We can also design features based on taxonomies to improve our models, such as the feature "*if two terms are assigned to the same parent node in a taxonomy*". Besides taxonomies or knowledge bases, query logs can also be helpful. They can be used to extract features to measure how useful and important facet

[2]www.freebase.com

terms or query facets are, and potentially improve term ranking within query facets, or facet ranking. For example, there may be many airlines extracted in the airline facet, based on statistics in query logs, we can easily find which airlines are more popular and rank them ahead in the query facet.

In Chapter 5, to improve the performance in precision-oriented scenarios, we used an empirical utility maximization approach that optimizes the performance measure for training our query faceting models. However, we have not investigated decision theoretic approaches, advocated by Lewis (1995), which try to optimize the performance measure during inferencing. Nan et al. (2012) compared both approaches for optimizing F-Measures. Their results suggest that the two approaches are asymptotically equivalent given large training and test sets. Nevertheless, their experiments show that the empirical utility maximization approach appears to be more robust against model misspecification. Given a good model, the decision-theoretic approach appears to be better for handling rare classes and a common domain adaptation scenario. It would be interesting to also develop decision theoretic approach for optimizing $PRF_{\alpha,\beta}$ measure, and compare with our proposed empirical utility maximization approach.

Lastly, in Chapter 7, we used a simple user interaction model based on some strong assumptions that may not hold in real scenarios. For example, in the user model, we originally model the time a user spends for scanning a facet F as $T(F)$. However, to simplify the estimation, we assumed time costs are equal for scanning different facets (*i.e.*, $T(F) = T_f$, where T_f is a constant). In reality, the time a user spends scanning a facet is highly dependent on the facet quality. Users may spend much more time for low-quality facets, in order to figure out what the facets are about. For example, the extracted facet {"*AA*", "*first*", "*Delta*", "*business*", "*JetBlue*"} mixes facet "*airline*" and facet "*flight classes*" together. When a user encounters this facet, he or she may

be confused and take more time in scanning. To improve time estimation, we could model the time cost based on the facet quality.

BIBLIOGRAPHY

Nasreen Abdul-Jaleel, James Allan, W Bruce Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark D Smucker, and Courtney Wade. UMass at TREC 2004: Novelty and hard. Technical report, DTIC Document, 2004.

Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 19–27. Association for Computational Linguistics, 2009.

Rakesh Agrawal, Sreenivas Gollapudi, Alan Halverson, and Samuel Ieong. Diversifying search results. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 5–14. ACM, 2009.

James Allan. Relevance feedback with too much data. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 337–343. ACM, 1995.

James Allan and Hema Raghavan. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 307–314. ACM, 2002.

Giambattista Amati, Claudio Carpineto, Giovanni Romano, and Fondazione Ugo Bordoni. Query difficulty, robustness, and selective application of query expansion. In *ECIR*, volume 4, pages 127–137. Springer, 2004.

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 588–596. Springer, 2004.

Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R Carvalho. Predicting query performance on the web. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 785–786. ACM, 2010.

Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *MACHINE LEARNING*, pages 238–247, 2002.

Francisco Barahona. On the computational complexity of ising spin glass models. *Journal of Physics A: Mathematical and General*, 15(10):3241, 1982.

Senjuti Basu Roy, Haidong Wang, Gautam Das, Ullas Nambiar, and Mukesh Mohania. Minimum-effort driven dynamic faceted search in structured databases. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 13–22. ACM, 2008.

Ori Ben-Yitzhak, Nadav Golbandi, Nadav Har'El, Ronny Lempel, Andreas Neumann, Shila Ofek-Koifman, Dafna Sheinwald, Eugene Shekita, Benjamin Sznajder, and Sivan Yogev. Beyond basic faceted search. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 33–44. ACM, 2008.

Sumit Bhatia, Debapriyo Majumdar, and Prasenjit Mitra. Query suggestions in the absence of query logs. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 795–804. ACM, 2011.

Dario Bonino, Fulvio Corno, and Laura Farinetti. Faset: A set theory model for faceted search. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 01*, pages 474–481. IEEE Computer Society, 2009.

Chris Buckley, Gerard Salton, James Allan, and Amit Singhal. Automatic query expansion using smart: TREC 3. *NIST special publication*, pages 69–69, 1995.

Robin D Burke, Kristian J Hammond, and Benjamin C Young. Knowledge-based navigation of complex information spaces. In *Proceedings of the National Conference on Artificial Intelligence*, volume 462, page 468, 1996.

Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

Robert G Capra and Gary Marchionini. The relation browser tool for faceted exploratory search. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 420–420. ACM, 2008.

Jaime Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 335–336. ACM, 1998.

Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. A survey of web clustering engines. *ACM Computing Surveys (CSUR)*, 41(3):17, 2009.

Ben Carterette and Praveen Chandar. Probabilistic models of ranking novel documents for faceted topic retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1287–1296. ACM, 2009.

Charles L Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2009 web track. Technical report, DTIC Document, 2009a.

Charles L Clarke, Nick Craswell, Ian Soboroff, and Gordon V Cormack. Overview of the TREC 2010 web track. Technical report, DTIC Document, 2009b.

Charles L Clarke, Nick Craswell, Ian Soboroff, and Ellen M Voorhees. Overview of the TREC 2011 web track. Technical report, DTIC Document, 2009c.

Charles L Clarke, Nick Craswell, and Ellen M Voorhees. Overview of the TREC 2012 web track. Technical report, DTIC Document, 2009d.

Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666. ACM, 2008.

Gordon V Cormack, Mark D Smucker, and Charles LA Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information retrieval*, 14(5): 441–465, 2011.

David Cossock and Tong Zhang. Subset ranking using regression. In *Learning theory*, pages 605–619. Springer, 2006.

Nick Craswell and Martin Szummer. Random walks on the click graph. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 239–246. ACM, 2007.

Steve Cronen-Townsend, Yun Zhou, and W Bruce Croft. Predicting query performance. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 299–306. ACM, 2002.

Steve Cronen-Townsend, Yun Zhou, and W Bruce Croft. A framework for selective query expansion. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 236–237. ACM, 2004.

Edward Cutrell, Daniel Robbins, Susan Dumais, and Raman Sarin. Fast, flexible filtering with phlat. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*, pages 261–270. ACM, 2006.

Douglass R Cutting, David R Karger, Jan O Pedersen, and John W Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM, 1992.

Wisam Dakka and Panagiotis G Ipeirotis. Automatic extraction of useful facet hierarchies from text databases. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 466–475. IEEE, 2008.

Wisam Dakka, Panagiotis G Ipeirotis, and Kenneth R Wood. Automatic construction of multifaceted browsing interfaces. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 768–775. ACM, 2005.

Van Dang and Bruce W Croft. Term level search result diversification. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 603–612. ACM, 2013.

Van Dang and W Bruce Croft. Diversity by proportionality: an election-based approach to search result diversification. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 65–74. ACM, 2012.

Van Dang, Xiaobing Xue, and W Bruce Croft. Inferring query aspects from reformulations using clustering. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2117–2120. ACM, 2011.

Debabrata Dash, Jun Rao, Nimrod Megiddo, Anastasia Ailamaki, and Guy Lohman. Dynamic faceted search for discovery-driven analysis. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 3–12. ACM, 2008.

Humberto Mossri de Almeida, Marcos André Gonçalves, Marco Cristo, and Pável Calado. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 399–406. ACM, 2007.

Melvil Dewey. *A Classification and Subject Index: For Cataloguing and Arranging the Books and Pamphlets of a Library*. Brick row book shop, Incorporated, 1876.

Mamadou Diao, Sougata Mukherjea, Nitendra Rajput, and Kundan Srivastava. Faceted search and browsing of audio content on spoken web. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1029–1038. ACM, 2010.

Zhicheng Dou, Sha Hu, Yulong Luo, Ruihua Song, and Ji-Rong Wen. Finding dimensions for queries. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1311–1320. ACM, 2011.

Jennifer English, Marti Hearst, Rashmi Sinha, Kirsten Swearingen, and Ka-Ping Yee. Hierarchical faceted metadata in site search interfaces. In *CHI'02 Extended Abstracts on Human Factors in Computing Systems*, pages 628–639. ACM, 2002.

Christiane Fellbaum. *WordNet. An electronic lexical database*. MIT Press, 1998.

Rowan Garnier and John Taylor. *Discrete mathematics: proofs, structures and applications*. CRC press, 2009.

Thomas R Gruber. A translation approach to portable ontology specifications. *Knowledge acquisition*, 5(2):199–220, 1993.

Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, 1954.

Marti Hearst. Design recommendations for hierarchical faceted search interfaces. In *ACM SIGIR workshop on faceted search*, pages 1–5. Seattle, WA, 2006a.

Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics, 1992.

Marti A. Hearst. Next generation web search: Setting our sites. *IEEE Data Eng. Bull.*, 23(3):38–48, 2000.

Marti A Hearst. Clustering versus faceted categories for information exploration. *Communications of the ACM*, 49(4):59–61, 2006b.

Marti A Hearst. Uis for faceted navigation: Recent advances and remaining open problems. In *Workshop on Computer Interaction and Information Retrieval, HCIR*, pages 13–17, 2008.

Marti A Hearst and Jan O Pedersen. Reexamining the cluster hypothesis: scatter/gather on retrieval results. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 76–84. ACM, 1996.

L.J. Heyer, S. Kruglyak, and S. Yooseph. Exploring expression data: identification and analysis of coexpressed genes. *Genome research*, 9(11):1106–1115, 1999.

Yunhua Hu, Yanan Qian, Hang Li, Daxin Jiang, Jian Pei, and Qinghua Zheng. Mining query subtopics from search log data. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 305–314. ACM, 2012.

Samuel Huston and W Bruce Croft. A comparison of retrieval models using term dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 111–120. ACM, 2014.

Alpa Jain and Gilad Mishne. Organizing query completions for web search. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 1169–1178. ACM, 2010.

Martin Jansche. Maximum expected f-measure training of logistic regression models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 692–699. Association for Computational Linguistics, 2005.

Yufeng Jing and W Bruce Croft. An association thesaurus for information retrieval. In *Proceedings of RIAO*, volume 94, pages 146–160, 1994.

Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

Rosie Jones, Benjamin Rey, Omid Madani, and Wiley Greiner. Generating query substitutions. In *Proceedings of the 15th international conference on World Wide Web*, pages 387–396. ACM, 2006.

Mika Käki. Findex: search result categories help users when document ranking fails. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 131–140. ACM, 2005.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

Jürgen Koenemann and Nicholas J Belkin. A case for interaction: a study of interactive information retrieval behavior and effectiveness. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 205–212. ACM, 1996.

Christian Kohlschütter, Paul-Alexandru Chirita, and Wolfgang Nejdl. Using link analysis to identify aspects in faceted web search. In *ACM SIGIR workshop on faceted search*, pages 55–59, 2006.

Weize Kong and James Allan. Extracting query facets from search results. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 93–102. ACM, 2013.

Weize Kong and James Allan. Extending faceted search to the general web. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 839–848. ACM, 2014.

Jonathan Koren, Yi Zhang, and Xue Liu. Personalized interactive faceted search. In *Proceedings of the 17th international conference on World Wide Web*, pages 477–486. ACM, 2008.

Zornitsa Kozareva, Ellen Riloff, and Eduard H Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL*, volume 8, pages 1048–1056, 2008.

Robert Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–202. ACM, 1993.

Bill Kules, Robert Capra, Matthew Banta, and Tito Sierra. What do exploratory searchers look at in a faceted search interface? In *Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries*, pages 313–322. ACM, 2009.

Ashish Kumar. A comparative analysis of taxonomy, thesaurus and ontology. *International Journal of Applied Services Marketing Perspectives*, 2(1):251, 2013.

KL Kwok, L Papadopoulos, and Kathy YY Kwan. Retrieval experiments with a large collection using pircs. In *The First Text REtrieval Conference (TREC-1)*, pages 153–172. US Department of Commerce Washington, DC, 1993.

Kui-Lam Kwok, Laszlo Grunfeld, HL Sun, Peter Deng, and N Dinstl. TREC 2004 robust track experiments using pircs. In *TREC*, 2004.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML '01*, pages 282–289, 2001. ISBN 1-55860-778-1.

K Latha, K Rathna Veni, and R Rajaram. AFGF: An automatic facet generation framework for document retrieval. In *Advances in Computer Engineering (ACE), 2010 International Conference on*, pages 110–114. IEEE, 2010.

Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 120–127. ACM, 2001.

Dawn Lawrie, W Bruce Croft, and Arnold Rosenberg. Finding topic words for hierarchical summarization. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 349–357. ACM, 2001.

Dawn J Lawrie and W Bruce Croft. Generating hierarchical summaries for web searches. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 457–458. ACM, 2003.

David D Lewis. Evaluating and optimizing autonomous text classification systems. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 246–254. ACM, 1995.

Chengkai Li, Ning Yan, Senjuti B Roy, Lekhendro Lisham, and Gautam Das. Facetedpedia: dynamic generation of query-dependent faceted interfaces for wikipedia. In *Proceedings of the 19th international conference on World wide web*, pages 651–660. ACM, 2010.

Dekang Lin. Automatic retrieval and clustering of similar words. In *Proceedings of the 17th international conference on Computational linguistics-Volume 2*, pages 768–774. Association for Computational Linguistics, 1998.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

Gang Luo, Chunqiang Tang, Hao Yang, and Xing Wei. Medsearch: a specialized search engine for medical information retrieval. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 143–152. ACM, 2008.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014.

Gary Marchionini and Ben Brunk. Towards a general relation browser: A gui for information architects. *Journal of Digital Information*, 4(1), 2003.

Qiaozhu Mei, Dengyong Zhou, and Kenneth Church. Query suggestion using hitting time. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 469–478. ACM, 2008.

Donald Metzler and W Bruce Croft. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 472–479. ACM, 2005.

Donald Metzler, W. Bruce Croft, and Andrew McCallum. Direct maximization of rank-based metrics for information retrieval. IR 429, University of Massachusetts, 2005.

David R Musicant, Vipin Kumar, Aysel Ozgur, et al. Optimizing f-measure with support vector machines. In *FLAIRS Conference*, pages 356–360, 2003.

Ye Nan, Kian Ming Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measure: A tale of two approaches. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 289–296, New York, NY, USA, July 2012. Omnipress. ISBN 978-1-4503-1285-1.

Craig G Nevill-Manning, Ian H Witten, and Gordon W Paynter. Lexically-generated subject hierarchies for browsing large collections. *International Journal on Digital Libraries*, 2(2-3):111–123, 1999.

Eyal Oren, Renaud Delbru, and Stefan Decker. Extending faceted navigation for rdf data. In *The Semantic Web-ISWC 2006*, pages 559–572. Springer, 2006.

Umut Ozertem, Emre Velipasaoglu, and Larry Lai. Suggestion set utility maximization using session logs. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 105–114. ACM, 2011.

Patrick Pantel and Dekang Lin. Discovering word senses from text. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–619. ACM, 2002.

Patrick Pantel, Deepak Ravichandran, and Eduard Hovy. Towards terascale knowledge acquisition. In *Proceedings of the 20th international conference on Computational Linguistics*, page 771. Association for Computational Linguistics, 2004.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. Web-scale distributional similarity and entity set expansion. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 938–947. Association for Computational Linguistics, 2009.

Marius Pasca. Acquisition of categorized named entities for web search. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 137–145. ACM, 2004.

Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. 1988. ISBN 0-934613-73-7.

Peter Pirolli, Stuart K Card, and Mija M Van Der Wege. The effect of information scent on searching information: visualizations of large tree structures. In *Proceedings of the working conference on Advanced visual interfaces*, pages 161–172. ACM, 2000.

A Steven Pollitt. The key role of classification and indexing in view-based searching. Technical report, 1998.

Wanda Pratt, Marti A Hearst, and Lawrence M Fagan. A knowledge-based approach to organizing retrieved documents. In *AAAI/IAAI*, pages 80–85, 1999.

C Quoc and Viet Le. Learning to rank with nonsmooth cost functions. *Proceedings of the Advances in Neural Information Processing Systems*, 19:193–200, 2007.

Shiyali Ramamrita Ranganathan. Colon clasification. 1933.

Shiyali Ramamrita Ranganathan. *Classification, coding and machinery for search*. Unesco, 1950.

J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The Smart retrieval system - experiments in automatic document processing*, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.

Kerry Rodden, Wojciech Basalaj, David Sinclair, and Kenneth Wood. Does organisation by similarity assist image browsing? In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 190–197. ACM, 2001.

Giovanni Maria Sacco and Yannis Tzitzikas. *Dynamic taxonomies and faceted search: theory, practice, and experience*, volume 25. Springer Science & Business Media, 2009.

Tetsuya Sakai and Ruihua Song. Evaluating diversified search results using per-intent graded relevance. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1043–1052. ACM, 2011.

Gerard Salton. Improving retrieval performance by relevance feedback. *Readings in information retrieval*, 24:5, 1997.

Gerard Salton, Edward A Fox, and Harry Wu. Extended boolean information retrieval. *Communications of the ACM*, 26(11):1022–1036, 1983.

Rodrygo LT Santos, Craig Macdonald, and Iadh Ounis. Exploiting query reformulations for web search result diversification. In *Proceedings of the 19th international conference on World wide web*, pages 881–890. ACM, 2010.

Shuming Shi, Xiaokang Liu, and Ji-Rong Wen. Pattern-based semantic class discovery with multi-membership support. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 1453–1454. ACM, 2008.

Shuming Shi, Huibin Zhang, Xiaojie Yuan, and Ji-Rong Wen. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 993–1001. Association for Computational Linguistics, 2010.

Keiji Shinzato and Kentaro Torisawa. A simple www-based method for semantic word class acquisition. *Recent Advances in Natural Language Processing IV: Selected Papers from RANLP 2005*, 292:207, 2005.

Ben Shneiderman. Dynamic queries for visual information seeking. *Software, IEEE*, 11(6):70–77, 1994.

David Smiley and David Eric Pugh. *Apache Solr 3 Enterprise Search Server*. Packt Publishing Ltd, 2011.

Rion Snow, Daniel Jurafsky, and Andrew Y Ng. Semantic taxonomy induction from heterogenous evidence. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 801–808. Association for Computational Linguistics, 2006.

R. Song, M. Zhang, T. Sakai, M.P. Kato, Y. Liu, M. Sugimoto, Q. Wang, and N. Orii. Overview of the NTCIR-9 intent task. In *Proceedings of NTCIR-9 Workshop Meeting*, pages 82–105, 2011.

Emilia Stoica and Marti A Hearst. Automating creation of hierarchical faceted metadata structures. In *Proceedings of NAACL-HLT*, 2007.

Bin Tan, Atulya Velivelli, Hui Fang, and ChengXiang Zhai. Term feedback for information retrieval with language models. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 263–270. ACM, 2007.

Jaime Teevan, Susan T Dumais, and Zachary Gutt. Challenges for supporting faceted search in large, heterogeneous corpora like the web. *Proceedings of HCIR*, 2008:87, 2008.

Stephen Tomlinson. Robust, web and terabyte retrieval with Hummingbird SearchServer^{TM} at TREC 2004. In *Proceedings of TREC*, 2004.

Daniel Tunkelang. Faceted search. *Synthesis lectures on information concepts, retrieval, and services*, 1(1):1–80, 2009.

Yannis Tzitzikas, Anastasia Analyti, and Nicolas Spyratos. Compound term composition algebra: the semantics. In *Journal on Data Semantics II*, pages 58–84. Springer, 2005.

Elizabeth Van Couvering. *The history of the Internet search engine: Navigational media and the traffic commodity*. Springer, 2008.

C.J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

Xuanhui Wang and ChengXiang Zhai. Learn from web search logs to organize search results. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 87–94. ACM, 2007.

Xuanhui Wang, Deepayan Chakrabarti, and Kunal Punera. Mining broad latent query aspects from search sessions. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 867–876. ACM, 2009.

Bifan Wei, Jun Liu, Qinghua Zheng, Wei Zhang, Xiaoyu Fu, and Boqin Feng. A survey of faceted search. *Journal of Web engineering*, 12(1-2):41–64, February 2013. ISSN 1540-9589.

Max L. Wilson and m.c. schraefel. Bridging the gap: Using ir models for evaluating exploratory search interfaces. In *SIGCHI 2007 Workshop on Exploratory Search and HCI*, 2007. Event Dates: 28th April 2007.

Fei Wu, Jayant Madhavan, and Alon Halevy. Identifying aspects for web-search queries. *Journal of Artificial Intelligence Research*, pages 677–700, 2011.

Jinxi Xu and W Bruce Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 4–11. ACM, 1996.

Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 391–398. ACM, 2007.

Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 107–114. ACM, 2008.

Xiaobing Xue and Xiaoxin Yin. Topic modeling for named entity queries. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 2009–2012. ACM, 2011.

Nan Ye, Kian Ming A Chai, Wee Sun Lee, and Hai Leong Chieu. Optimizing f-measures: a tale of two approaches. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Ka-Ping Yee, Kirsten Swearingen, Kevin Li, and Marti Hearst. Faceted metadata for image search and browsing. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 401–408. ACM, 2003.

Xiaoxin Yin and Sarthak Shah. Building taxonomy of web search intents for name entity queries. In *Proceedings of the 19th international conference on World wide web*, pages 1001–1010. ACM, 2010.

Elad Yom-Tov, Shai Fine, David Carmel, and Adam Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 512–519. ACM, 2005.

Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278. ACM, 2007.

Oren Zamir and Oren Etzioni. Web document clustering: A feasibility demonstration. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 46–54. ACM, 1998.

Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to web search results. *Computer Networks*, 31(11):1361–1374, 1999.

Cheng Xiang Zhai, William W Cohen, and John Lafferty. Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 10–17. ACM, 2003.

Huibin Zhang, Mingjie Zhu, Shuming Shi, and Ji-Rong Wen. Employing topic models for pattern-based semantic class discovery. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 459–467. Association for Computational Linguistics, 2009.

Junliang Zhang and Gary Marchionini. Evaluation and evolution of a browse and search interface: Relation browser++. In *Proceedings of the 2005 national conference on Digital government research*, pages 179–188. Digital Government Society of North America, 2005.

Junliang Zhang, Gary Marchionini, Tim Shear, and Chang Su. Relational browser: A fast and contextualized searching and browsing tool. Technical report, 2004.

Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 363–370. ACM, 2010.

Yun Zhou and W Bruce Croft. Ranking robustness: a novel framework to predict query performance. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 567–574. ACM, 2006.